An Efficient Approach to Pruning Regression Trees Using a Modified Bayesian Information Criterion

by

Nikola Surjanovic

B.Sc., Simon Fraser University, 2015

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
Department of Statistics and Actuarial Science
Faculty of Science

SIMON FRASER UNIVERSITY
Spring 2021

Declaration of Committee

Name:	**Nikola Surjanovic**
Degree:	**Master of Science (Statistics)**
Thesis title:	**An Efficient Approach to Pruning Regression Trees Using a Modified Bayesian Information Criterion**
Committee:	**Chair:** Joan Hu Professor, Statistics and Actuarial Science
	Thomas M. Loughin Supervisor Professor, Statistics and Actuarial Science
	Richard Lockhart Committee Member Professor, Statistics and Actuarial Science
	Lloyd T. Elliott Examiner Assistant Professor, Statistics and Actuarial Science

Abstract

By identifying relationships between regression tree construction and change-point detection, we show that it is possible to prune a regression tree efficiently using properly modified information criteria. We prove that one of the proposed pruning approaches that uses a modified Bayesian information criterion consistently recovers the true tree structure provided that the true regression function can be represented as a subtree of a full tree. In practice, we obtain simplified trees that can have prediction accuracy comparable to trees obtained using standard cost-complexity pruning. We briefly discuss an extension to random forests that prunes trees adaptively in order to prevent excessive variance, building upon the work of other authors.

Keywords: Regression trees; pruning; information criteria; cross-validation; machine learning

Acknowledgements

I first have to thank my supervisor, Dr. Tom Loughin, for his constant support over the past several years while I was completing my undergraduate and master's degrees at SFU. Tom has taught me a great deal about how to think like a scientist and how to communicate ideas clearly. His great sense of humour has also made completing a master's degree enjoyable. He has definitely set a great example for me as to what a supervisor should be.

There are too many other members of the department to thank. I would like to express appreciation towards Dr. Rachel Altman and Dr. Derek Bingham for warmly welcoming me to the department back in 2015. Conversations with Dr. Richard Lockhart in the past few years have always been extremely interesting and insightful. Many thanks to all members of the committee for reading my thesis, including Dr. Lloyd Elliott, who also kindly served as a mentor for a case competition last year. To all members of the department: I will definitely miss SFU.

I also have to thank my family and friends. I am grateful for the love and support of my parents, Ivan and Yatsa. A big thank-you to my older sister, Sonja, for always leading the way and setting a good example. Many thanks to all of my friends for the nice memories.

Table of Contents

List of Figures

Chapter 1

Introduction

Predicting the value of a numeric response variable from a given set of explanatory variables is central to virtually every scientific discipline. This can be achieved using one of many available regression methods. In this work we will focus on regression trees, a popular type of statistical learning algorithm, because they are often used as the foundation for more complex learning algorithms.

The basic premise behind regression trees is relatively simple. Here we present only an overview and provide more details in Section 2.1. Tree construction is based on a recursive partitioning (RP) or "splitting" of subsets of the explanatory variable space in order to form a partition of the entire space. This process continues until some termination criterion is attained. After a partition is obtained, the regression tree algorithm estimates the regression surface by taking the mean of the observed response values in each of the regions. The obtained trees are therefore piecewise constant regression functions and the regions can be obtained through a sequence of simple rules that can be summarized as a tree diagram. An example of a tree diagram and the corresponding estimated surface are given in Figures 2.1 and 2.2.

Our work focuses on developing methods for "pruning" regression trees efficiently and providing theoretical performance guarantees of these methods. Pruning of a regression tree refers to the process of simplifying the tree in order to offer better predictions and/or ease of interpretation. The trade-off between bias and variance when pruning a tree is best illustrated by considering two extreme examples: a tree in which no splits are made and all predictions for future observations are exactly the same, and a tree in which each observation in the training set forms its own group so that means for prediction in each group are estimated using only one response. While hardly any of the features of the true regression function can be explained by the first tree (high bias) if the true regression function is complex, refitting the same tree on a new training set would barely cause the tree to change (low variance). On the other hand, the average of many highly complex trees produced from independent datasets might approximate the true regression function reasonably well (low bias), but a new set of training data would drastically change the shape of the estimated

regression function obtained from a single tree (high variance). Clearly, there may be a tree that lies between these two extremes that is optimal (in some sense) for prediction. Often, large regression trees are built initially and a pruning algorithm is used to find the best tree, taking into consideration the bias-variance trade-off.

The standard pruning algorithm relies on cross-validation, which can be a computationally expensive procedure. We develop alternative pruning algorithms that are computationally efficient and can easily be extended to tree-based ensemble learning algorithms such as random forests. The proposed algorithms use a modified Bayesian information criterion during the pruning process. We obtain simplified trees that have prediction accuracy often comparable to trees obtained using standard pruning. In some cases, our trees are better in terms of prediction accuracy. We also offer some theoretical guarantees on the performance of one of the proposed algorithms. An extension to random forests that prevents the growth of trees with excessive variance, suggested by [9], is briefly discussed.

In Chapter 2, we review the RP tree-construction algorithm of [1], information criteria, and the change-point detection problem. Then, in Chapter 3, we outline the standard pruning algorithm, along with other existing pruning algorithms, and introduce the proposed pruning algorithms based on a modified Bayesian information criterion. The theoretical consistency results are also provided in this chapter. The design of the simulation study used to evaluate the performance of the pruning methods is outlined in Chapter 4 and the results are provided in Chapter 5. We conclude with a brief discussion, along with some suggestions for future work with tree-based ensemble learning algorithms, in Chapter 6. The proofs of several results can be found in the Appendix.

Chapter 2

Literature Review

In this chapter, we first review the recursive partitioning (RP) algorithm of Breiman et al. [1] for tree construction. This algorithm, described in Section 2.1, is a very commonly used method for constructing classification and regression trees via a RP procedure. Next, in Section 2.2, we give a brief overview of various information criteria, including the Bayesian information criterion and Akaike's information criterion. We end the literature review with a summary of various relevant papers on change-point detection in Section 2.3. The process of fitting a regression tree is essentially a recursively applied change-point detection procedure, and work in this research area with information criteria forms the basis of our pruning procedure.

Throughout this thesis, we write $(\mathbf{X}_1, Y_1), (\mathbf{X}_2, Y_2), \ldots$ to denote a sequence of i.i.d. pairs of random variables. The random variable $\mathbf{X}_i = (X_i^{(1)}, X_i^{(2)}, \ldots, X_i^{(p)})'$ takes on values in $\mathcal{X} \subset \mathbb{R}^p$ and represents a vector of explanatory variables, while the random variable Y_i takes on values in $\mathcal{Y} \subset \mathbb{R}$ and is the response. For a given sample size n, we use lowercase to denote realizations of these random variables: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$.

2.1 Regression Trees and the CART Algorithm

We lay out the RP algorithm used by Breiman et al. [1], which we refer to as the CART (classification and regression tree) construction algorithm, combining notation from [1] and [22]. We also introduce some new notation that will be useful for describing our own pruning algorithms in Section 3.2.

The CART algorithm provides a way to estimate the regression function by partitioning the explanatory variable space and taking the mean of the observed responses in each region to form a piecewise-constant estimated regression function. During tree construction, CART provides a systematic way of partitioning $\mathcal{X}$ by choosing appropriate splitting hyperplanes and regions on which to split. At each step, CART searches for a split that reduces the sum of squared errors the most. Given a region $A \subset \mathcal{X}$, $j \in \{1, 2, \ldots, p\}$, and $z \in \mathbb{R}$, we define $A_L = \{\mathbf{x} \in A : x^{(j)} < z\}$ to be the "left" side of A, and $A_R = \{\mathbf{x} \in A : x^{(j)} \geq z\}$ to be the

"right" side of A. We also let $\mathbb{1}(S)$ be the indicator function on a set S, and write $|S|$ to denote the cardinality of a finite set S. For an arbitrary region A we define

$$\bar{Y}_{n,A} = \sum_{i=1}^{n} Y_i \mathbb{1}(\mathbf{X}_i \in A) / \left| \sum_{i=1}^{n} \mathbb{1}(\mathbf{X}_i \in A) \right|$$

and set this variable equal to zero if the numerator and denominator are both zero. Finally, we define the random quantity

$$\begin{aligned} L_{n,A}(j,z) = & \sum_{i=1}^{n} (Y_i - \bar{Y}_{n,A})^2 \mathbb{1}(\mathbf{X}_i \in A) \\ & - \left[\sum_{i=1}^{n} (Y_i - \bar{Y}_{n,A_L})^2 \mathbb{1}(\mathbf{X}_i \in A_L) + \sum_{i=1}^{n} (Y_i - \bar{Y}_{n,A_R})^2 \mathbb{1}(\mathbf{X}_i \in A_R) \right], \end{aligned} \quad (2.1)$$

which is the reduction in sum of squares after splitting on region A on the jth variable at location z and fitting a separate mean in each of the new partition regions.

Starting with the region $A = \mathcal{X}$ and the partition $\mathcal{P}_0 = \{\mathcal{X}\}$, the CART algorithm selects the values of j and z so that (2.1) is *maximized*. Because the value of z that maximizes this quantity is not unique, it is customary to set it equal to a value halfway between two consecutive values of the $x_i^{(j)}$. The optimal pair (j, z) defines a splitting hyperplane, H_1, and a new partition $\mathcal{P}_1 = \{A_L, A_R\}$. The procedure is then continued within each element of $\mathcal{P}_1$. For each partition, the elements of the partition are referred to as nodes. If a node is split on with a hyperplane, it becomes the parent of two child nodes.

At some point, depending on the stopping rule used, the algorithm is terminated and a partition $\mathcal{P}_K$ of $\mathcal{X}$ containing $K + 1$ elements (or "terminal nodes") is obtained. From this partition, we can estimate the regression function with

$$m_{n,K}(\mathbf{x}) = \sum_{A \in \mathcal{P}_K} \bar{Y}_{n,A} \mathbb{1}(\mathbf{x} \in A) \quad (2.2)$$

for $\mathbf{x} \in \mathcal{X}$. Given a training set $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, n$, the non-random estimated regression function, or fitted "tree", can be used to predict responses for new observations. An example regression tree, represented both as a graph and as a surface, is presented in Figures 2.1 and 2.2.

There are many implementations of the CART tree-construction procedure, some of which can be performed with the `rpart()` function from the `rpart` package in R. Implementations can vary, depending on the tree-growing rules and stopping rules used. For example, trees can be grown in a balanced manner, ensuring that the depth of each tree branch is roughly equal. Alternatively, the trees can be grown "best first", meaning that there is no restriction on tree balance and that all terminal nodes are eligible for splitting in a given step of the tree construction algorithm. Various parameters can be set, such as the

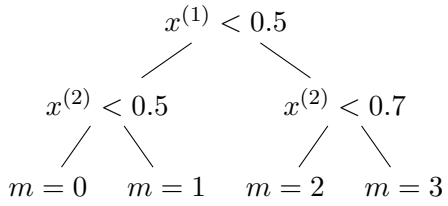

Figure 2.1: An example of a simple regression tree represented as a graph. Proceed to the left child node if a condition is met. Otherwise, proceed to the right child node. m denotes the estimated regression function value in the given region.

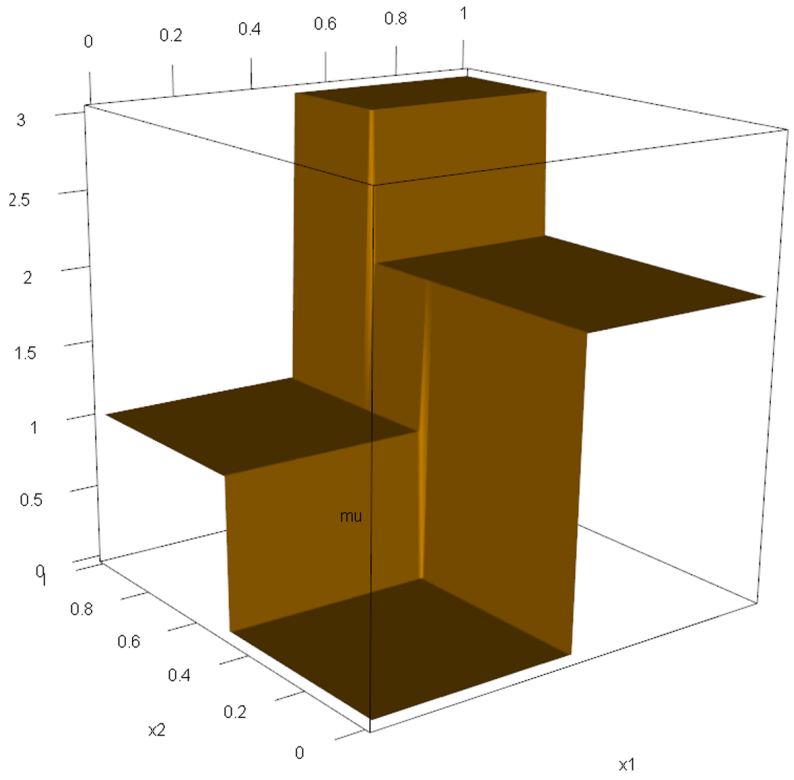

Figure 2.2: The regression tree from Figure 2.1 represented as a surface.

minimum number of observations required to make a split on a particular node (`minsplit`) and the minimum number of observations that can be in a terminal node (`minbucket`). Unless otherwise stated, we assume that splits are allowed to occur on nodes with as few as four observations (`minsplit=4`) and we must have at least two observations in each terminal node (`minbucket=2`).

2.2 Information Criteria

In this section we describe several information criteria used for model selection. Our new pruning algorithms make use of a modified version of the Bayesian information criterion (BIC) [21] and so we focus on this criterion. However, we also try using a modification of Akaike's information criterion (AIC) and examine its performance.

2.2.1 Bayesian Information Criterion

The BIC [21], sometimes referred to as Schwarz's information criterion, is a tool for model selection. Despite its name, its use is not limited to Bayesian models. It is only the derivation of the information criterion that is based on Bayesian principles. In what follows, we introduce the BIC in a way similar to [11]. The BIC is quite general and its derivation is not necessarily limited to models of the type to be presented.

Suppose that we have D models: $M_1, M_2, \ldots, M_D$. For any one of these D models, say model d, we have an associated set of distributions f_{θ_d} that are indexed by some parameter θ_d in $\Theta_d \subset \mathbb{R}^{p_d}$ for some p_d. Further, model d specifies a prior, π_d, on θ_d. That is, given model d, we have

$$\begin{aligned} \theta_d | M_d &\sim \pi_d, \\ \mathbf{X}_1, \ldots, \mathbf{X}_n | M_d, \theta_d &\overset{iid}{\sim} f_{\theta_d}. \end{aligned}$$

We let π_M denote the prior on the D considered models, so that $\pi_M(M_d)$ is the prior probability that the dth model is the "true" model. From here, we can obtain the posterior probabilities for each of the models conditional on the observed data, $\mathbf{x}_1, \ldots, \mathbf{x}_n$. Applying Bayes' rule,

$$P(M = M_d | \mathbf{X}_1 = \mathbf{x}_1, \ldots, \mathbf{X}_n = \mathbf{x}_n) = \frac{\left[\int_{\Theta_d} \left(\prod_{i=1}^n f_{\theta_d}(x_i)\right) \pi_d(d\theta_d)\right] \pi_M(M_d)}{P(\mathbf{X}_1 = \mathbf{x}_1, \ldots, \mathbf{X}_n = \mathbf{x}_n)}. \tag{2.3}$$

The quantity $P(\mathbf{X}_1 = \mathbf{x}_1, \ldots, \mathbf{X}_n = \mathbf{x}_n)$ is used to represent a density if the $\mathbf{X}_i$ are continuous, for example.

Intuitively, it seems like a good idea to select the model with the highest posterior probability. If π_M is uniform across the D models, then this is the same as choosing the

model with the highest value of

$$p_d(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \int_{\Theta_d} \left(\prod_{i=1}^{n} f_{\theta_d}(x_i) \right) \pi_d(d\theta_d), \tag{2.4}$$

a quantity often referred to as the marginal likelihood. In contrast, the log-likelihood given model d evaluated at an arbitrary parameter θ is $\ell_{n,d}(\theta) = \sum_{i=1}^{n} \log(f_\theta(\mathbf{x}_i))$ and we write $\hat{\theta}_{n,d}$ to refer to the maximum likelihood estimate (MLE) of θ_d for model d.

By using some approximations, such as the Laplace approximation, one can show that (under some regularity conditions)

$$-2\log(p_d(\mathbf{x}_1, \ldots, \mathbf{x}_n)) \approx -2\ell_{n,d}(\hat{\theta}_{n,d}) + p_d \log(n).$$

The BIC for model d is defined as

$$\mathrm{BIC}_n(M_d) = -2\ell_{n,d}(\hat{\theta}_{n,d}) + p_d \log(n), \tag{2.5}$$

and we choose the model with the *smallest* BIC value, thereby choosing the model that yields approximately the largest marginal likelihood.

Remarkably, the approximation in (2.5) does not depend on the priors π_d and contains quantities that are often quite easily computed. Furthermore, under certain conditions, it can be shown that the BIC selects the "true" model with probability approaching one as the sample size goes to infinity.

In Section 2.3 we describe models in which the BIC from (2.5) is not appropriate due to the presence of parameters that are not regular. However, it is still possible to create an information criterion for such models by properly accounting for these parameters.

2.2.2 Akaike's Information Criterion

The AIC approaches model selection from a different angle: instead of selecting the model with the highest posterior probability, it searches for the model with the greatest prediction accuracy. This is done by searching for a model with the smallest expected Kullback-Leibler divergence relative to the true model. We present the AIC in a way similar to [4] and [11]. Again, the AIC is not necessarily limited to the models of the type that we will consider.

Let Z be independent of the training data X but with the same distribution, f, as X. The discussions in [4] and [11] explain that minimizing the expected value of the Kullback-Leibler divergence of the estimated model relative to the true model is equivalent to maximizing

$$\mathrm{E}(\log f(Z|\hat{\theta}_{n,d}(X))), \tag{2.6}$$

the expectation of the log-likelihood taken over X and Z.

To illustrate the importance of using (2.6) as a model selection criterion compared to the maximized likelihood alone, consider a collection of models, $M_1, M_2, \ldots, M_D$, as before. Suppose that these models are now *nested* and we use the maximized log-likelihood to choose from them. Selecting the model with the largest log-likelihood will always favor the most complex model, simply because $\ell(\hat{\theta}_{n,d_1}) \leq \ell(\hat{\theta}_{n,d_2})$ for $d_1 \leq d_2$, because $\Theta_{d_1} \subset \Theta_{d_2}$ in this case. The reason for this behaviour is that the maximized log-likelihood is a biased estimate of (2.6) and a penalty is needed that properly accounts for the model size.

We have already seen that the BIC introduces a penalty on model dimension, introducing a trade-off between maximized likelihood and model size. The AIC does the same, but with a different penalty. By making asymptotic approximations to quantities related to (2.6), the information criterion in this framework is

$$AIC_n(M_d) = -2\ell_n(\hat{\theta}_{n,d}) + 2p_d. \tag{2.7}$$

The AIC penalty is smaller than the BIC penalty provided that $n \geq 8$, and it therefore tends to favour larger models than the BIC. In general, the AIC is not guaranteed to select the "true" model with probability approaching one, in contrast to the BIC.

2.2.3 Other Information Criteria

There is an abundance of other information criteria, including the Focused Information Criterion [3], the Deviance Information Criterion (DIC) [25, 26], and the Widely Applicable (Bayesian) Information Criterion [30], among others. A good review of AIC, BIC, DIC, and some other information criteria can be found in [11]. These information criteria each have their own merits. However, we will not consider information criteria beyond AIC and BIC any further.

2.3 Change-Point Detection

For the remainder of this chapter, we focus on change-point detection. Although the literature on change-point detection and change-point regression is vast, we focus on change-point detection methods that rely on the use of information criteria. The regression tree construction algorithm is essentially a recursive change-point detection procedure and we would like to prune regression trees using information criteria.

Consider a sequence of independent (but not necessarily identically distributed) random variables $Y_1, Y_2, \ldots, Y_n$. In the simplest setting, one might wish to determine whether the mean of the responses changes at some point along the sequence. If the mean does change at some point, we say that there is a change-point present. Otherwise, there is no change-point (assuming that only the mean can change). In the case of a single change-point, we might also wish to determine its location. Formally, we wish to know whether there is a k such

that $1 \leq k < n$ and

$$\mathrm{E}(Y_1) = \cdots = \mathrm{E}(Y_k) \neq \mathrm{E}(Y_{k+1}) = \cdots = \mathrm{E}(Y_n).$$

In the single change-point setting, a natural interpretation of this model is that a certain process was used to generate the data up to time k, at which a change occurred and the rest of the data was obtained using a different process.

Regression tree construction is essentially a recursive change-point detection procedure with two main differences. The first difference between tree construction and change-point detection is that the order of the sequence of responses is determined by the order of one of the explanatory variables and so there may be several possible orderings to consider. Second, if the explanatory variables are random, the ordering of the responses can also be random.

The standard method used to obtain a BIC for regular models does not apply in the change-point setting because change-point problems contain parameters that are not regular (the location of the change-point). To address this problem, Shen and Ghosh [24] develop a modified BIC for change-point problems. Their BIC is "a lower bound to the marginal likelihood of a model with change points and has an approximation error up to $O_p(1)$ like standard Schwartz BIC" [24]. An important result is that a change-point parameter should be treated as two regular parameters. We let M_0 and M_1 denote the zero and single change-point models, respectively, and write $\hat{k}_n$ for the estimated change-point location under M_1. We also write $\hat{\theta}_{n,0}$ and $\hat{\theta}_{n,1}$ to denote the estimates of the mean(s) in the zero and single change-point models so that $p_0 = 1$ and $p_1 = 2$. Using $lBIC$ to denote their "lower bound" BIC (up to a multiplicative factor of -2), we have

$$lBIC_n(M_0) = -2\ell_n(\hat{\theta}_{n,0}) + p_0 \log(n), \tag{2.8}$$

$$lBIC_n(M_1) = -2\ell_n(\hat{\theta}_{n,1}, \hat{k}_n) + (p_1 + 2)\log(n). \tag{2.9}$$

In their analysis, [24] restrict attention to a certain family of distributions, of which the normal distribution with unknown mean and known variance is a member. For our purposes, we will restrict our attention to this single-parameter normal distribution and have $p_0 = 1$ and $p_1 = 2$. The case with unknown variance is described in detail in Section 3.2. The analysis in [24] also includes a discussion on multiple change-points, but we focus only on the single change-point problem.

Other change-point papers that make use of information criteria include [31, 16, 8, 17, 12]. Ninomiya [16, 17] concludes that with the AIC, change-point parameters should be

treated as three regular parameters. In our context,

$$NAIC_n(M_0) = -2\ell_n(\hat{\theta}_{n,0}) + 2p_0 \tag{2.10}$$

$$NAIC_n(M_1) = -2\ell_n(\hat{\theta}_{n,1}, \hat{k}_n) + 2(p_1 + 3), \tag{2.11}$$

where $p_0 = 1$, $p_1 = 2$, and we write $NAIC$ to refer to "Ninomiya's AIC" used in [16, 17]. Both the $lBIC$ and $NAIC$ will be used with our proposed pruning algorithms, to be laid out in the next chapter.

Chapter 3

Methods

In this chapter we first present the standard pruning algorithm of [1]. We then develop several new pruning algorithms to be used in our simulation study of Chapter 4. We also present some other existing pruning algorithms for both classification and regression trees and state a consistency result for one of our proposed pruning algorithms.

3.1 Cost-Complexity Pruning

The cost-complexity pruning algorithm presented by Breiman et al. [1] is a very commonly used algorithm for classification and regression tree pruning. After obtaining a large tree using the CART tree construction algorithm, the cost-complexity pruning algorithm—hereafter referred to as the CART pruning algorithm—considers a sequence of nested subtrees and selects the one with the best estimated error rate using cross-validation.

Suppose that we have used the CART tree-construction algorithm to obtain a (full) regression tree, T_K, after making K splits. The tree T_K contains $K+1$ terminal nodes and K internal nodes. Formally, T_K is a graph with labelled nodes that provide instructions on how to create the estimated regression function, $m_{n,K}$, displayed in (2.2).

We define a subtree T_k of T_K to be a tree with $k+1 \leq K+1$ terminal nodes that shares the same root node with T_K and has branches that extend to any of the internal or terminal nodes of T_K. With this definition, the full tree T_K is considered a subtree of itself. For any two arbitrary trees T and T', if T' is a subtree of T, we write $T' \preccurlyeq T$.

We would like to use the CART pruning algorithm to find a subtree $T_k \preccurlyeq T_K$ that has the "best" prediction accuracy. For a given subtree T_k of T_K with $k+1$ terminal nodes and corresponding estimated regression function $m_{n,k}$, we define the (training set) sum of squared errors as

$$SSE(T_k) = \sum_{i=1}^{n} (y_i - m_{n,k}(\mathbf{x}_i))^2.$$

Because T_k is estimated using the training data, $SSE(T_k)$ is a biased estimate of the expected sum of squared errors on test data. To account for this bias, the CART pruning

algorithm introduces a penalized sum of squared errors that adds a penalty term that is proportional to the number of terminal nodes, $k+1$. For a given $\alpha \in [0, \infty)$, define

$$SSE_\alpha(T_k) = SSE(T_k) + \alpha(k+1).$$

From here, the algorithm constructs a sequence of $q \leq K+1$ nested subtrees

$$T_0 = T_{k_1} \preccurlyeq T_{k_2} \preccurlyeq \cdots \preccurlyeq T_{k_q} = T_K.$$

The number of subtrees in the sequence, q, depends on some factors such as the structure of the full tree T_K. The sequence is obtained by increasing α, starting from zero, and finding the subtree of T_K that minimizes SSE_α. Note that when $\alpha = 0$, the best tree is the most complex tree, which is T_K itself. On the other hand, as α goes to infinity, terminal nodes are penalized more heavily and the best tree becomes the one with a single terminal node, T_0, the root of the original tree.

Because α is a tuning parameter, the CART pruning algorithm typically finds the optimal value of α by cross-validation (CV), although a separate test set can also be used. For each CV fold, we fit a regression tree on the training data in that fold, prune for various values of α, and evaluate the test error using the CV test data. The optimal value of α is estimated to be the one that yields the lowest average test error across the CV folds. The full tree is then pruned using this estimated value of α.

3.2 BIC Pruning

Having laid out the standard regression tree pruning algorithm, we now propose two BIC-based pruning algorithms: BIC pruning with "branch freezing" (BIC-BF) and BIC pruning with "accumulated information" (BIC-AI), which borrows the concept of accumulated information from [9]. We provide an overview of the algorithms and offer a more detailed description of the algorithms in Sections 3.2.2 and 3.2.3. Starting from the terminal nodes last created during tree construction, pairs of nodes obtained from a common split are merged or retained according to a modified BIC. If the difference in means of the observed responses in the two nodes is not sufficiently large according to the criterion, the two nodes are merged and a new terminal node is obtained. On the other hand, if the criterion indicates that the difference in means is sufficiently large, then one of two approaches is taken depending on whether we are using the BIC-BF or BIC-AI algorithm.

With branch freezing (BIC-BF), the split (and hence terminal nodes) are kept and the branch containing the two nodes is "frozen". Then, node merges in other parts of the tree are considered. This process is repeated until either only the root node is left or all of the branches are frozen. The details are provided in Section 3.2.2.

With accumulated information (BIC-AI), if higher parts of the same branch are later determined to be filled with noise and low information, the entire branch may be removed at a later point during the pruning process. More details are given in Section 3.2.3.

Both BIC-BF and BIC-AI rely only on quantities that can be easily stored during the tree construction process, and so pruning can be done quickly in one pass through the tree after the full tree is constructed. We will also see that a simpler version of these algorithms is able to recover the true underlying tree structure under certain conditions.

3.2.1 Notation

We first clarify the notation used in the BIC-based pruning algorithms. For a fixed sample size n, $(A_{0,n}, H_{1,n}, \ldots, A_{K-1,n}, H_{K,n})$ denotes a sequence of nodes and splitting hyperplanes (perpendicular to one of the axes) that induce a *random* partition $\mathcal{P}_{K,n}$ with $K+1$ terminal nodes. Any given hyperplane acts on the region that appears immediately before it in the sequence. For example, the tree in Figures 2.1 and 2.2 can be represented by the sequence $(A_0, H_1, A_1, H_2, A_2, H_3)$, where $A_0 = [0,1] \times [0,1]$, $A_1 = [0.5,1] \times [0,1]$, $A_2 = [0,0.5] \times [0,1]$, H_1 splits on $X^{(1)}$ at 0.5, H_2 splits on $X^{(2)}$ at 0.7, and H_3 splits on $X^{(2)}$ at 0.5.

Such a sequence and partition can be obtained by using a tree construction algorithm and terminating after exactly K splits have been made. For BIC pruning, we use a "best first" tree construction approach so that at each step of tree construction we select the split that reduces the sum of squared errors the most *among all current terminal nodes*. The subscripts on the regions and hyperplanes indicate the exact order in which nodes are selected and splits are made.

For a region A and hyperplane H, we define $\text{Cut}(A, H)$ to be the partition of A that is induced by the hyperplane H. If $H = \{\mathbf{x} \in \mathcal{X} : x^{(j)} = z\}$ for some j, z, then

$$\text{Cut}(A, H) = \{\{\mathbf{x} \in A : x^{(j)} < z\}, \{\mathbf{x} \in A : x^{(j)} \geq z\}\}.$$

We generalize the definition of $lBIC$ from (2.8) and (2.9) to the p-dimensional "tree stump" problem. Suppose, for the moment, that $\mathbf{X} \in [0,1]^p$ has a continuous distribution and let A be a closed, axis-aligned rectangular subset of $[0,1]^p$. Let

$$N_A = \sum_{i=1}^{n} \mathbb{1}(\mathbf{X}_i \in A),$$

the number of observations in the region (a random quantity), and let

$$\bar{Y}_A = \sum_{i=1}^{n} Y_i \mathbb{1}(\mathbf{X}_i \in A)/N_A,$$

the mean response value in the given region (where $0/0 = 0$). When the region A is clear from the context, the subscript is omitted from both of these quantities. From here, if σ^2 is

known, we define

$$lBIC_{0,N_A}(A) = N_A \log(2\pi\sigma^2) + \frac{1}{\sigma^2}\sum_{i=1}^{n}(Y_i - \bar{Y}_A)^2 \mathbb{1}(\mathbf{X}_i \in A) + \log(N_A). \tag{3.1}$$

This is the $lBIC$ evaluated for the "no split" regression tree model. If σ^2 is *unknown* and replaced with an estimate $\hat{\sigma}^2$, we define

$$lBIC_{0,N_A}(A, \hat{\sigma}^2) = N_A \log(2\pi\hat{\sigma}^2) + \frac{1}{\hat{\sigma}^2}\sum_{i=1}^{n}(Y_i - \bar{Y}_A)^2 \mathbb{1}(\mathbf{X}_i \in A) + 2\log(N_A). \tag{3.2}$$

(A penalty is added for the variance estimate even if it is not the MLE. A description of possible estimators is given in Sections 3.2.2 and 3.2.3 and an explanation is given in the Appendix in Lemma 4.) If $A = [0,1]^p$, then

$$lBIC_{0,n}([0,1]^p, \hat{\sigma}^2) = n\log(2\pi\hat{\sigma}^2) + \frac{1}{\hat{\sigma}^2}\sum_{i=1}^{n}(Y_i - \bar{Y})^2 + 2\log(n).$$

The $lBIC$ of the "no split" model is then compared to the $lBIC$ of the "single split" model with a split-point estimated by the CART tree-construction algorithm. To precisely state the $lBIC$ for the "single split" or "tree stump" model, we introduce some further notation. For convenience, we assume that $A = [0,1]^p$, but the quantities to be defined can easily be generalized to other regions. Let $(\mathbf{X}_1^{*j}, Y_1^{*j}), \ldots, (\mathbf{X}_n^{*j}, Y_n^{*j})$ be the permutation of $(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)$ such that the j-th coordinates of the $\mathbf{X}_i$ are sorted by increasing value. Then, for $n_L = 1, \ldots, n-1$, define

$$\begin{aligned}\bar{Y}_{L,n_L}^{*j} &= \frac{Y_1^{*j} + \ldots + Y_{n_L}^{*j}}{n_L}, \\ \bar{Y}_{R,n_L}^{*j} &= \frac{Y_{n_L+1}^{*j} + \ldots + Y_n^{*j}}{n - n_L},\end{aligned}$$

the "left" and "right" averages of the responses, respectively, for a split on $X^{(j)}$ that places n_L observations in the left node. Finally, we define

$$\begin{aligned}&lBIC_{1,n}([0,1]^p, \hat{\sigma}^2) \\ &= n\log(2\pi\hat{\sigma}^2) + \min_{\substack{n_L=1,\ldots,n-1 \\ j=1,\ldots,p}} \frac{1}{\hat{\sigma}^2}\left[\sum_{i=1}^{n_L}(Y_i^{*j} - \bar{Y}_{L,n_L}^{*j})^2 + \sum_{i=n_L+1}^{n}(Y_i^{*j} - \bar{Y}_{R,n_L}^{*j})^2\right] + 5\log(n).\end{aligned} \tag{3.3}$$

The definition can easily be extended to a closed, rectangular region $A \subset [0,1]^p$ in which not all n responses are observed. If σ^2 is *known*, then the penalty is $4\log(n)$ intead of $5\log(n)$.

3.2.2 Branch Freezing (BIC-BF)

We present the proposed BIC pruning algorithm for regression trees with branch freezing, which we refer to as BIC-BF. An input to the algorithm is the ordered sequence of regions and hyperplanes that produce the full tree, $(A_0, H_1, \ldots, A_{K-1}, H_K)$. The notation is explained in Section 3.2.1. The order of the nodes being based on "best first" tree construction is important because we will traverse the nodes/regions A in the reverse order (skipping some if a branch is frozen during pruning), starting with A_{K-1}.

Algorithm 1: BIC-BF: BIC bottom-up pruning algorithm with branch freezing

Input: $\mathcal{P}_{K,n}$, $(A_{0,n}, H_{1,n}, \ldots, A_{K-1,n}, H_{K,n})$, $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$
Output: $\mathcal{P}^{BIC}$
$\mathcal{P}^{BIC} \leftarrow \mathcal{P}_{K,n}$;
$R \leftarrow (A_{0,n}, A_{1,n}, \ldots, A_{K-1,n})$; # Regions to test
while $R \neq \emptyset$ **do**
  $A \leftarrow$ last element of R;
  if $lBIC_{1,n}(A, \hat{\sigma}^2) < lBIC_{0,n}(A, \hat{\sigma}^2)$ **then**
    # Do not merge: freeze the branch
    $R \leftarrow \{B \in R : A \text{ is not a subset of } B\}$;
  else
    # Merge nodes
    $\mathcal{P}^{BIC} \leftarrow \mathcal{P}^{BIC} \setminus \text{Cut}(A_{K-k,n}, H_{K-k+1,n}) \cup \{A_{K-k,n}\}$;
    $R \leftarrow R \setminus \{A\}$;
  end
end
return $\mathcal{P}^{BIC}$

In practice, σ^2, which is used in the calculation of the generalized $lBIC$, is unknown and should be replaced with an estimator. Our approach in the BIC-BF algorithm is to use a different estimate of σ^2 at each pruning step by calculating the average sum of squared errors in the two terminal nodes under consideration. This means that the same estimate is used in $lBIC_{1,n}(A, \hat{\sigma}^2)$ and $lBIC_{0,n}(A, \hat{\sigma}^2)$ and it is the MLE from the former "single split" model. Alternatively, one could replace the estimate of σ^2 in the latter "no split" model with its own MLE, but this does not seem to be necessary in light of Lemma 4, which suggests that one can use the same estimate of σ^2 in all $lBIC$s during the *entire* pruning process (for the BIC-ET algorithm to be discussed in Section 3.5).

When the algorithm determines that two terminal nodes have sufficiently different means according to the modified BIC, the branch containing the two terminal nodes is frozen. The pruning continues by moving to another branch that has not yet been frozen and the algorithm terminates when either all branches are frozen or only the root node remains.

3.2.3 Accumulated Information (BIC-AI)

We now present an alternative BIC-based pruning algorithm that employs a less strict stopping rule than BIC-BF. The pruning method is based on the concept of "accumulated information" from [9] and we refer to the algorithm as BIC-AI. Instead of freezing a branch when two nodes cannot be merged, the pruning process continues to climb up the tree. It is possible that these two nodes, along with several other nodes, become collapsed into a single node if it is later determined that their means are collectively not too different from one another.

We introduce the concept of accumulated information in a way similar to [9], borrowing some notation used there. Let n_A be the number of $\mathbf{x}_i$ in a given region A, the observed value of the random quantity N_A.

Suppose that the tree to be pruned was constructed by splitting on the regions/nodes $A_0, A_1, \ldots, A_{K-1}$ in this *exact* order using the "best first" construction approach, as explained in Section 3.2.1. We traverse these nodes and make pruning decisions in exactly the *reverse* order, beginning with the node A_{K-1}. We begin by defining the quantity I_A for each node A of the tree to be missing. For a node A that is currently under consideration, with children A_L and A_R, if I_{A_L} and/or I_{A_R} are missing, we set the missing values to the (almost-maximized) log-likelihood of the data in the region multiplied by -2:

$$I_{A_L} \leftarrow n_{A_L} \log(2\pi\hat{\sigma}^2) + \frac{1}{\hat{\sigma}^2} \sum_{i=1}^{n} (y_i - \bar{y}_{A_L})^2 \mathbb{1}(\mathbf{x}_i \in A_L), \tag{3.4}$$

$$I_{A_R} \leftarrow n_{A_R} \log(2\pi\hat{\sigma}^2) + \frac{1}{\hat{\sigma}^2} \sum_{i=1}^{n} (y_i - \bar{y}_{A_R})^2 \mathbb{1}(\mathbf{x}_i \in A_R).$$

It is important to note that the *same* estimate $\hat{\sigma}^2$ is used if both I_{A_L} and I_{A_R} are missing. The estimate $\hat{\sigma}^2$ that is used is the average of the squared errors from the subtree that has a root at A and extends to include all nodes directly below A that have not yet been pruned from the tree. That is, $\hat{\sigma}^2$ is the weighted average of the estimated variances in each terminal node below A, with weights equal to the number of observations in each terminal node. This ensures that for a tree stump with only two terminal nodes, $I_{A_L} + I_{A_R}$ yields the *maximized* log-likelihood (multiplied by -2) based on the "single split" homoscedastic model. Because A has not yet been traversed, I_A is necessarily missing and we assign it the value

$$I_A \leftarrow n_A \log(2\pi\hat{\sigma}^2) + \frac{1}{\hat{\sigma}^2} \sum_{i=1}^{n} (y_i - \bar{y}_A)^2 \mathbb{1}(\mathbf{x}_i \in A),$$

where $\hat{\sigma}^2$ is the *same* as for A_L and/or A_R. (This means that I_A is not exactly equal to -2 times the *maximized* log-likelihood for the "no split" model for observations in A. By Lemma 4, we can use the same estimate in all $lBIC$s during the *entire* pruning process for BIC-ET, a simplified pruning algorithm discussed in Section 3.5. For BIC-AI we update $\hat{\sigma}^2$

in each pruning step for each new node that is traversed, but still apply the same estimate for any of I_{A_L}, I_{A_R}, and I_A that are yet to be defined in the current pruning step.)

To assess whether to merge the subtrees rooted at A_L and A_R into the single node A, we add penalties to each of I_{A_L}, I_{A_R}, and I_A, exactly as in the $lBIC$. If

$$I_{A_L} + I_{A_R} + 5\log(n_A) < I_A + 2\log(n_A), \tag{3.5}$$

the split is kept. Otherwise, the subtrees rooted at A_L and A_R are merged into a single terminal node. For a tree stump, this is equivalent to checking whether $lBIC_{1,n_A}(A, \hat{\sigma}^2) < lBIC_{0,n_A}(A, \hat{\sigma}^2)$.

If we choose to keep a split, the algorithm does not freeze the entire branch. If we keep the split on node A, we define

$$I_A \leftarrow I_{A_L} + I_{A_R} + 3\log(n_A), \tag{3.6}$$

subtracting $2\log(n_A)$ from $5\log(n_A)$ to avoid double-counting parameters in future applications of (3.5). The pruning algorithm continues to climb up the tree, updating the values of I_A in regions A that have yet to be traversed. For example, if the split on the region A_{K-1} was just assessed, we then proceed to assess the split on the region A_{K-2}, and so on.

If at any point the comparison in (3.5) is false and we choose to merge the nodes, the parent node A becomes a new terminal node with a missing value of I_A.

The BIC-AI pruning algorithm is very similar to the one proposed by [9], but there are three main differences: a modified BIC is used instead of an AIC, penalty values do not need to be drawn from a table based on simulated values, and it is applied to trees with homoscedastic responses (instead of heteroscedastic responses).

3.3 AIC Pruning

It is possible to construct AIC-based analogues of the pruning methods from the previous section by replacing the modified BIC with a modified AIC in which split-point parameters are counted as three regular parameters, as in [16, 17]. This yields the pruning methods AIC-BF (branch freezing) and AIC-AI (accumulated information). Analogous to the $NAIC$ in (2.10) and (2.11), the penalties $2\log(n)$ and $5\log(n)$ are replaced with 4 and 12 for the "no split" and "single split" models, respectively. We introduce them mainly to examine whether they have the ability to create trees with good prediction accuracy (because this is what the AIC is designed for) and to compare them to the BIC pruning methods and cost-complexity pruning. The pruning methods with a modified AIC should yield larger trees than the BIC-based methods, because the penalty is smaller in the AIC for sufficiently large n.

3.4 Other Pruning Methods

Although the CART pruning algorithm seems to be the main method for pruning trees, there is a wealth of literature on alternative pruning algorithms. In this section, we review some of these other existing methods. We start off with a description of pruning algorithms for regression that make use of information criteria or related measures. After presenting these, we introduce other pruning algorithms for regression and classification trees.

3.4.1 IC-Based Pruning Methods

Several methods for pruning regression trees using information criteria have been proposed. For example, [2] introduced regression trees that can be used with various response types. One of the proposed pruning algorithms first creates a sequence of subtrees and then compares these trees using a version of the AIC that does not account for split-point parameter estimation. In contrast, our method prunes bottom-up without creating a sequence of subtree candidates and makes use of a modified BIC (that accounts for split-point parameters) instead of an AIC.

While not a pruning method in the traditional sense, [7] use a genetic algorithm to construct regression trees using BIC as a measure of the quality of the fit of a tree. However, they use the number of terminal nodes plus one as the number of parameters in the BIC penalty. They acknowledge that other researchers have "suggested that the effective number of parameters estimated is actually much higher ... due to split rule selections made during the tree construction process", and conclude that "further research is required to determine the appropriate adjustment of the model complexity penalty term in the BIC criterion". Our results suggest that split-point parameters should be treated as two regular parameters in the BIC, as was obtained for the change-point problem by [24].

Recently, [14] suggested a top-down tree construction algorithm using information criteria that allows one to construct a gradient boosted tree ensemble in an automated way. By focusing on a single tree in the ensemble, one obtains a construction algorithm that terminates as soon as a certain condition is met, instead of constructing a full tree and then pruning. This algorithm is exciting, but we feel that post-pruning (rather than pre-pruning) allows one to consider a larger number of candidate trees and may be better for individual trees or ensembles with deep trees such as random forests (we discuss an extension of our pruning approach to random forests in Chapter 6).

[19] introduced CORE, a regression tree algorithm that can prune trees using minimum description length (MDL). The CORE algorithm allows for general types of models in the terminal nodes. An example of a pruning algorithm for *regression* trees using MDL can be found in [20]. These authors acknowledge that MDL has been used for pruning *decision* trees previously, but provide a method for pruning when responses are numeric.

A type of regression tree algorithm based on maximum likelihood was proposed by [27]. Their pruning algorithm makes use of (a choice between) AIC, BIC, and two other information criteria. However, the information criteria do not seem to account for the estimation of split-points in the regression tree construction algorithm, and the authors additionally use a test set to estimate the sum of squares present in the likelihood. In our opinion, this approach is slightly counter-intuitive, because the AIC penalty is specifically designed to account for the bias present in likelihood estimates obtained using a training set.

3.4.2 Other Pruning Methods for Regression Trees

There are, of course, regression tree pruning algorithms that do not rely on information criteria. One such example is that of [13], who considered a shrinkage method that also prunes trees. [18] proposed the M5 algorithm, which fits linear models in tree nodes. While pruning bottom-up with this algorithm, an estimate of the test error is obtained by multiplying the training error by a coefficient that depends on the sample size and the number of parameters. According to [28], [10] prunes using a type of estimator known as an m-estimator. In their case, the m-estimator is used to obtain a weighted average of the prior and posterior estimates of the error. Each of these pruning algorithms is interesting, but we feel that there is room for our pruning approach that makes use of a modified BIC and possesses some desirable consistency properties.

3.4.3 Other Pruning Methods for Classification Trees

In contrast to regression trees, classification trees seem to have received a considerable amount of attention for the development of pruning algorithms. Because the focus of this work is on regression tree pruning, we do not provide many details here. A good review of some of these pruning algorithms is given by [5, 15]. It is interesting to note that [6] present a way to classify various pruning algorithms by considering four key properties of the algorithm that they refer to as "space", "operators", "evaluation function", and "search strategy". It might be of interest to categorize various existing regression tree pruning algorithms using their system.

3.5 BIC Pruning: Consistency Results

In this section we present a consistency result for a simpler version of the BIC-BF pruning algorithm. The modified algorithm terminates earlier than BIC-BF and we therefore refer to the algorithm as BIC-ET (early termination). BIC-ET is simpler because as soon as we make the first decision not to merge any two nodes, the algorithm terminates completely. In order to guarantee consistency, the algorithm also discards observations that lie too close to the estimated split-point locations.

A consistency proof for BIC-ET can likely be created for BIC-BF with some small modifications, but we leave this extension for future work. Importantly, a generalization for BIC-BF could allow for a considerable weakening of some of the assumptions on the tree construction algorithm. For example, with BIC-ET, we need to assume that there is a point in the construction algorithm at which we start to make only unnecessary splits and assume that all preceding splits are correct. A consistency result for BIC-BF might be able to weaken this assumption considerably.

Some more definitions are needed in order to precisely state the BIC-ET algorithm. Some of the conditions and definitions in this section are similar to or have analogues to those presented in [22].

We first introduce the notion of a tree-based partition of the explanatory variable space. Tree-based partitions are partitions that can be obtained using a splitting procedure such as the CART algorithm.

Definition 1. *We say that a partition $\mathcal{P}$ is a tree-based partition if there exists a sequence of nodes and hyperplanes (not necessarily unique), $A_0 = \mathcal{X}, H_1, A_1, H_2, \ldots, A_{k-1}, H_k$, that induces this partition (using the notation described in Section 3.2.1).*

Consider a tree-based partition $\mathcal{P}$ of $[0,1]^p$. For any $A \in \mathcal{P}$, A is necessarily of the form

$$A = [a_1, b_1) \times \cdots \times [a_p, b_p),$$

for some constants $a_j, b_j \in [0,1]$. (The right bracket above is closed for any $b_j = 1$.)

Next, we introduce the notion of an ϵ-reduction of a region or a partition. These reductions are used in the BIC-ET algorithm to discard observations that lie too close to region boundaries in order to ensure the consistency of the pruning algorithm. An example of an ϵ-reduction is illustrated in Figure 3.1.

Definition 2. *For $\epsilon > 0$, we define the ϵ-reduction of A to be the closed set*

$$A(\epsilon) = [a_1 + \epsilon, b_1 - \epsilon] \times \cdots \times [a_p + \epsilon, b_p - \epsilon],$$

provided that ϵ is small enough for the quantity to be defined. Otherwise, we set $A(\epsilon) = \emptyset$. We define the ϵ-reduction of the partition $\mathcal{P}$ as

$$\mathcal{P}(\epsilon) = \{A(\epsilon) : A \in \mathcal{P}\}.$$

Our final definition before stating the BIC-ET algorithm introduces the "Snap" operation applied to a given region. It is similar to an ϵ-reduction, except that it ensures that the vertices of a reduced rectangular region lie exactly on some predefined grid of points. The use of the "Snap" makes the proof of BIC-ET consistency relatively simple, because it

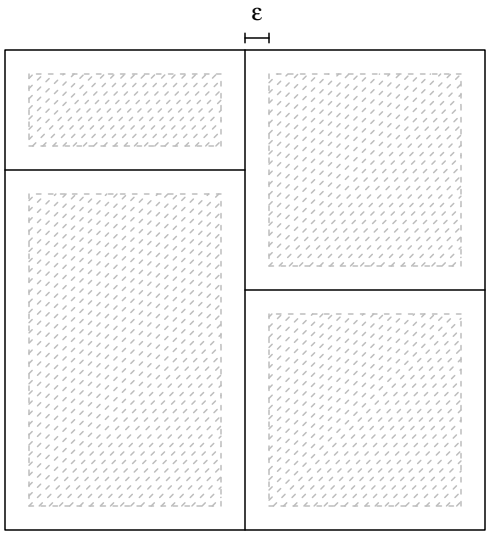

Figure 3.1: A tree-based partition, $\mathcal{P}$, of $[0,1]^2$. The grey rectangles form the ϵ-reduction of $\mathcal{P}$.

ensures that there are only finitely many possible "snapped" regions on $[0,1]^p$. An example is given in Figure 3.2.

Definition 3. *For any $\epsilon > 0$, define $\mathcal{G}(\epsilon)$ to be the grid of points $\{0, 1/\lceil 1/\epsilon \rceil, 2/\lceil 1/\epsilon \rceil, \ldots, 1\}$, so that the points are equally spaced and the distance between any two consecutive points is less than or equal to ϵ. Given a region A, define*

$$\text{Snap}(A, \mathcal{G}(\epsilon)) = \{\textit{The largest } B \subset A : B = [a_1', b_1'] \times \cdots \times [a_p', b_p']; a_j', b_j' \in \mathcal{G}(\epsilon)\}.$$

(As before, if there is no such region B, then we set $\text{Snap}(A, \mathcal{G}(\epsilon)) = \emptyset$*.)*

For the BIC-ET algorithm, we also need to input the quantities σ^2, $\tilde{\epsilon}$, and N. For the consistency proof of Theorem 1, we assume the true value of σ^2 is known, but we show in the Appendix in Lemma 4 that we can replace σ^2 with a weakly consistent estimator $\hat{\sigma}_n^2$. The last two quantities $\tilde{\epsilon}$ and N should be chosen to satisfy some properties that will be given in detail in Section 3.5.1. For now, $\tilde{\epsilon}$ should simply be viewed as a very small number greater than zero and N as a very large number. In practice, these two quantities should not impact the performance of BIC-ET but are primarily used to prove consistency of the pruning algorithm.

Now that we have introduced some definitions, we can state the BIC-ET pruning algorithm precisely in Algorithm 2. As with BIC-BF, an input to the algorithm is the ordered sequence of regions and hyperplanes that produce the full tree, $(A_0, H_1, \ldots, A_{K-1}, H_K)$. The nodes are ordered according to "best first" tree construction, as explained in Section 3.2.1 and they are traversed in the reverse order, starting with A_{K-1}.

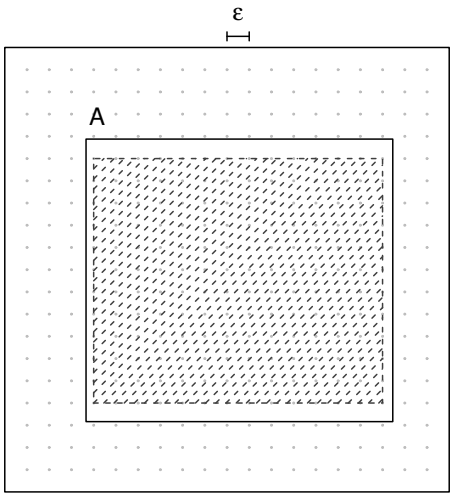

Figure 3.2: A region $A \subset [0,1]^2$ with $\text{Snap}(A, \mathcal{G}(\epsilon))$ displayed as a shaded grey rectangle. The light grey points form a grid on $[0,1]^2$ to which regions can be "snapped".

Algorithm 2: BIC-ET: BIC bottom-up pruning algorithm with early termination

Input: $\mathcal{P}_{K,n}$, $(A_{0,n}, H_{1,n}, \ldots, A_{K-1,n}, H_{K,n})$, $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, $\tilde{\epsilon}$, N, σ^2
Output: $\mathcal{P}^{BIC}$
$k \leftarrow 1$;
$\mathcal{P}^{BIC} \leftarrow \mathcal{P}_{K,n}$;
flag $\leftarrow$ TRUE;
while $k \leq K$ *AND flag* **do**
  $A \leftarrow \text{Snap}(A_{K-k,n}(\tilde{\epsilon}), \mathcal{G}(\tilde{\epsilon}/N))$;
  Restrict attention to observations in A and estimate the location of a split-point within A. (If A has fewer observations than $A_{K-k,n}$, the split-point will need to be estimated again.);
  if $lBIC_{1,n}(A) < lBIC_{0,n}(A)$ **then**
    # Do not merge, end procedure
    flag $\leftarrow$ FALSE;
  else
    # Merge nodes
    $\mathcal{P}^{BIC} \leftarrow \mathcal{P}^{BIC} \setminus \text{Cut}(A_{K-k,n}, H_{K-k+1,n}) \cup \{A_{K-k,n}\}$;
    $k \leftarrow k + 1$;
  end
end
return $\mathcal{P}^{BIC}$

3.5.1 Theorem and Lemmas: Definitions, Conditions, and Statements

Having stated the BIC-ET algorithm, we state two lemmas and a theorem that establish the consistency result for BIC-ET. Some additional definitions and conditions are now presented in order to state the results precisely.

Definitions

We start off by defining tree-producible functions, which are piecewise constant on regions that can be obtained with a tree construction algorithm. BIC-ET recovers the correct number of terminal nodes in a regression tree as well as the approximate structure of the true tree. It does not make much sense to refer to the "correct" number of terminal nodes if the true regression function cannot be represented as a tree.

Definition 4. *A regression function $m : \mathcal{X} \to \mathbb{R}$ is said to be tree-producible if it can be written in the form*

$$m(\mathbf{x}) = \sum_{A \in \mathcal{P}} c_A \mathbb{1}(\mathbf{x} \in A)$$

for some tree-based partition $\mathcal{P}$ and some constants $c_A \in \mathbb{R}$.

Next, we introduce the concept of a partition "refinement". The partitions of the explanatory variable space obtained by a tree construction algorithm after split K are always more "refined" than the partitions obtained after split $k < K$. An example is illustrated in Figure 3.3.

Definition 5. *Consider two tree-based partitions: $\mathcal{P}_1$ and $\mathcal{P}_2$. If for every $A_2 \in \mathcal{P}_2$ there is a set $A_1 \in \mathcal{P}_1$ such that $A_2 \subset A_1$, then we say that $\mathcal{P}_2$ is a refinement of $\mathcal{P}_1$ and we write $\mathcal{P}_1 \preccurlyeq \mathcal{P}_2$.*

Intuitively, $\mathcal{P}_1 \preccurlyeq \mathcal{P}_2$ means that one can combine the elements of $\mathcal{P}_2$ to reconstruct $\mathcal{P}_1$. This is essentially what pruning does to the terminal nodes of a regression tree. Note that the refinement symbol, $\preccurlyeq$, includes equality.

The next definition, which introduces partition perturbations, is used to quantify the amount of uncertainty present in the recovered "true" subtree after pruning. Even if the tree-construction algorithm selects the correct variables to split on in the correct order, the estimated split-point location is unlikely to be exactly correct. A perturbation of a partition is one in which the regions have been "squished" or "stretched" without changing the underlying structure (correct splitting variables) of the original partition (see Figure 3.4).

Definition 6. *Consider a tree-based partition $\mathcal{P}$ of $[0,1]^p$ and a region $A \in \mathcal{P}$. Given another region $A' \subset [0,1]^p$ of the form $[a'_1, b'_1) \times \cdots [a'_p, b'_p)$ (the right bracket can be closed if $b_j = 1$), and a constant $\epsilon \geq 0$, we say that A' is an ϵ-perturbation of A if*

$$\max_j |a'_j - a_j| \text{ and } \max_j |b'_j - b_j| \leq \epsilon.$$

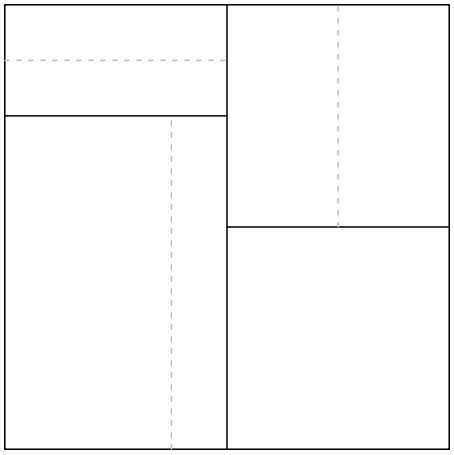

Figure 3.3: A partition $\mathcal{P}$ (rectangles with black borders) of $[0,1]^2$ and a possible refinement of $\mathcal{P}$ (additional dashed grey lines).

Given another partition $\mathcal{P}'$, we say that $\mathcal{P}'$ is an ϵ-perturbation of $\mathcal{P}$ if, for any $A' \in \mathcal{P}'$, there is an $A \in \mathcal{P}$ such that A' is an ϵ-perturbation of A, and $|\mathcal{P}| = |\mathcal{P}'|$.

The final definition makes it clear that it may be the case that more than one sequence of nodes and hyperplanes can produce *exactly* the same tree-based partition. This definition allows us to acknowledge the fact that the tree-construction algorithm might arrive at the same tree through various sequences of variable splits (although this should be unlikely). For example, consider the partition of $[0,1]^2$ containing four squares of equal size. It is unclear (and irrelevant) whether this partition was obtained by splitting on $X^{(1)}$ first or by splitting first on $X^{(2)}$.

Definition 7. *Let $\mathcal{P}$ be a tree-based partition of $[0,1]^p$ and suppose that $|\mathcal{P}| = K+1$. Define*

$$\text{Gen}(\mathcal{P}) = \{(A_0 = [0,1]^p, H_1, A_1, \ldots, A_{K-1}, H_K) \text{ that generate } \mathcal{P}\}.$$

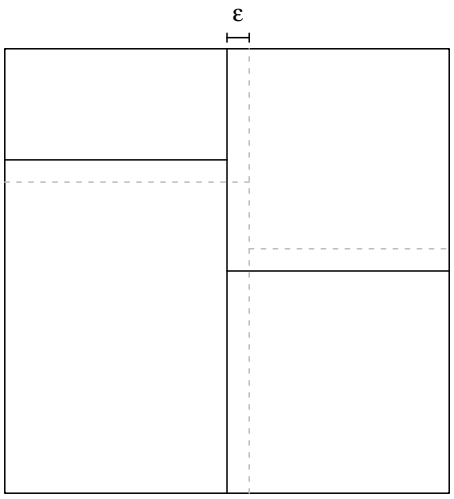

Figure 3.4: A partition $\mathcal{P}$ (rectangles with black borders) of $[0,1]^2$ and a possible ϵ-perturbation of $\mathcal{P}$ (dashed grey lines).

Conditions

There are three types of conditions. An interpretation of the conditions of type (C) is given below. Note that not all conditions of a given type are always assumed. Instead, this is a list of conditions that appear at some point in one or more of the lemmas or the main theorem.

Conditions of Type A (Distributional Assumptions)

1. $\mathbf{X}_1$ is a continuous random variable with strictly positive density on $[0,1]^p$ with respect to Lebesgue measure. The density is zero elsewhere.

2. $Y_1|\mathbf{X}_1 = \mathbf{x}_1 \sim N(m(\mathbf{x}_1), \sigma^2)$ for some $\sigma^2 > 0$.

Conditions of Type B (Regression Function Assumptions)

1. m is tree-producible from $\mathcal{P}$, a tree-based partition induced by a single hyperplane perpendicular to the standard basis vector $\mathbf{e}_{j_0}$, but is not tree-producible from $\mathcal{P}_0 = \{[0,1]^p\}$. That is, m is a tree "stump". This implies that $\mu_{1,0} \neq \mu_{2,0}$, where $\mu_{1,0} = \mathrm{E}(Y_1|X_1^{(j_0)} < z_0)$ and $\mu_{2,0} = \mathrm{E}(Y_1|X_1^{(j_0)} \geq z_0)$.

2. The true value of z, z_0, the location of the split along variable j_0, lies in the open interval $(0,1)$.

3. For some $K_\mu > 0$, $m(\mathbf{x}) \in (-K_\mu, K_\mu)$ for all $\mathbf{x} \in [0,1]^p$.

4. m is tree-producible from $\mathcal{P}$ but is not tree-producible from any other partition $\mathcal{P}'$ with $|\mathcal{P}'| < |\mathcal{P}|$.

Conditions of Type C (Partition Assumptions)

1. For some random ϵ_n, $\mathcal{P}_{K,n}$ is a random refinement of an ϵ_n-perturbation of $\mathcal{P}$, $\mathcal{P}_n^*$, which is itself random.

2. $|\mathcal{P}_{K,n}| = K + 1$ for all n (K is fixed).

3. There exists a *fixed* K^* such that $K^* \leq K$ and $(A_{0,n}, H_{1,n}, \ldots, A_{K^*-1,n}, H_{K^*,n}) \in \text{Gen}(\mathcal{P}_n^*)$ a.s.

4. There exists some $\epsilon > 0$, such that $P(\epsilon_n < \epsilon) \to 1$. Let $M_n = \sup\{\epsilon : \emptyset \notin \mathcal{P}_{K,n}(\epsilon)\}$. Further assume that there exists a constant $\tilde{\epsilon} > \epsilon$ such that $P(\epsilon < \tilde{\epsilon} < M_n) \to 1$.

5. There exists an N such that $P(\text{Snap}(A_{K-k,n}(\tilde{\epsilon}), \mathcal{G}(\tilde{\epsilon}/N)) \neq \emptyset) \to 1$ for all $1 \leq k \leq K - K^* + 1$.

6. The probability that $\text{Snap}(A_{K^*-1,n}(\tilde{\epsilon}), \mathcal{G}(\tilde{\epsilon}/N))$ contains exactly one split-point that is in the interior of this region approaches one as $n \to \infty$.

Conditions of types (A) and (B) are relatively easy to interpret. Condition (C1) says that the tree-building procedure selects the right splits in approximately the correct locations, but also makes some unnecessary additional splits. (C2) says that the tree-building procedure terminates after exactly K splits have been made. (C3) states that the true tree contains exactly $K^* + 1$ terminal nodes and that the true underlying partition can be recovered by eliminating the last $K - K^*$ splits in exactly the reverse order of tree-building. (C4) is a statement on the convergence of the split-point estimators that is similar to convergence in probability, and (C5) says that two split-point locations are not "too close" to one another and can be enforced, for example, by setting a constraint on how close estimated split points are allowed to be during tree construction. (C4) and (C5) should be satisfied, loosely speaking, if the estimators of the split-points are weakly consistent. The last condition, (C6), roughly says that split-points do not occur on estimated region boundaries. If we were to know the exact split-point locations, we could try to avoid this by choosing $\tilde{\epsilon}$ and N so that the "Snap" of a region never has a split-point exactly on the region boundary.

Theorem and Lemma Statements

We introduce two lemmas and a theorem. Lemma 1 is *not* used in Theorem 1, but is useful for convincing oneself that the assumptions of condition set (C) are reasonable. Lemma 2 forms the basis of Theorem 1, which precisely states the consistency of the BIC-ET algorithm. In general, the proofs provided in this thesis should be considered as sketches and a work in progress.

Lemma 1 states that the estimator of the split-point is consistent when there is a single split in the regression function (a tree "stump"). To introduce Lemma 1, we first define the

quantity

$$M_n(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \mu_1 \mathbb{1}(X_i^{(j)} < z) - \mu_2 \mathbb{1}(X_i^{(j)} \geq z)\right)^2,$$

where $\theta = (j, z, \mu_1, \mu_2)'$.

Lemma 1. *Consider a regression function $m : [0,1]^p \to \mathbb{R}$, where $p \geq 1$. Suppose that conditions (A1)-(A2) and (B1)-(B3) are satisfied. Define*

$$\hat{\theta}_n = \arg\max_{\theta \in \Theta} M_n(\theta),$$

where $\Theta = \{1, 2, \ldots, p\} \times [0,1] \times [-K_\mu, K_\mu] \times [-K_\mu, K_\mu]$. The maximum can clearly be attained (i.e., it exists), but to ensure uniqueness, let $\hat{\theta}_n = (j_n, z_n, \mu_{1,n}, \mu_{2,n})'$ be the value with z_n as the "midpoint" of the first interval among the set of z that maximize the likelihood, for example. If there are ties among the possible estimates of j, j_n, take the minimum value.

Then,

$$\hat{\theta}_n \xrightarrow{p} \theta_0 = (j_0, z_0, \mu_{1,0}, \mu_{2,0})'.$$

Consequently, $P(j_n = j_0) \to 1$ and $z_n \xrightarrow{p} z_0$.

From this lemma, conditions (C4) and (C5) (consistency of estimated split-point locations) should seem reasonable assuming that the tree-construction algorithm selects the right variables to split on. With a large enough sample size, our estimate of the split-point location should be close to the true split-point location with high probability.

Lemma 2 states that the proposed $lBIC$ from (3.1) and (3.3) selects the appropriate model (either "no split" or "single split") asymptotically when one of these is true. *This version of the lemma assumes that σ^2 is known.* For the case where σ^2 is replaced with a consistent estimator $\hat{\sigma}_n^2$, see Lemma 4 in the Appendix.

Lemma 2. *Consider a regression function $m : [0,1]^p \to \mathbb{R}$, where $p \geq 1$. Suppose that conditions (A1)-(A2) are satisfied. From here, we split the lemma into two cases.*

Case 1: *If, in addition to the conditions above, m is equal to μ_0 on $[0,1]^p$, for some constant $\mu_0 \in (-K_\mu, K_\mu)$ with $K_\mu > 0$, then*

$$P(lBIC_{1,n}([0,1]^p) > lBIC_{0,n}([0,1]^p)) \to 1$$

as $n \to \infty$. That is, the probability of selecting the model with no splits approaches one.

Case 2: *If instead, in addition to conditions (A1)-(A2), conditions (B1)-(B3) are satisfied, then,*

$$P(lBIC_{0,n}([0,1]^p) > lBIC_{1,n}([0,1]^p)) \to 1$$

as $n \to \infty$. That is, the probability of selecting the model with a single split approaches one.

Finally, the main theorem states that BIC-ET recovers the true underlying tree structure, including the correct number of terminal nodes, with a small amount of uncertainty in the exact split-point locations.

Theorem 1. *Suppose that conditions (A1)-(A2), (B3)-(B4) and (C1)-(C6) are satisfied. Let* $\mathcal{P}_n^{BIC}$ *be the random partition obtained using BIC-ET with sample size* n*. Then,*

$$P(\mathcal{P}_n^{BIC} = \mathcal{P}_n^*) \to 1$$

as $n \to \infty$*.*

The proof of the theorem relies on a small extension of Lemma 2 that remains to be proven. A comment is provided in the proof of the theorem in the Appendix.

Chapter 4

Simulation Study Design

In this chapter we lay out the design of our simulation study, which is used to assess the performance of several pruning algorithms from Chapter 3. We are interested in two properties of the pruning algorithms: their ability to recover the correct model dimension (number of terminal nodes) and the prediction accuracy of the pruned trees.

For the first property, model dimension recovery, we wish to assess whether our proposed BIC-based pruning algorithms, BIC-AI and BIC-BF, can consistently recover the correct number of terminal nodes, as suggested by Theorem 1 for BIC-ET. To determine the proportion of times that the correct number of terminal nodes is recovered under various circumstances, we consider three different settings each with multiple sub-settings. The settings are obtained by varying the distributions, the underlying tree structure, the sample size, and the maximum number of splits in the regression tree construction algorithm.

The second property, prediction accuracy, is important because the pruning algorithms are often used before making predictions (either with single trees or in ensembles such as a random forest). To study prediction accuracy, we use the same three settings as for model dimension recovery and include an additional setting: one in which the true underlying regression function cannot be represented as a tree.

Unless otherwise stated, the explanatory variables $\mathbf{X}_1, \ldots, \mathbf{X}_n$ are i.i.d. and drawn from a uniform distribution on $[0,1]^p$. The responses $Y_1, \ldots, Y_n$ are independent and normally distributed with variance σ^2 and a mean that is dependent on the true underlying regression function. Due to the current implementation of the algorithms, if $x_1^{(j)}, \ldots, x_n^{(j)}$ are not all unique for any j, the observations are drawn again. We write p and p_0 to denote the number of variables available and the true number of variables related to the response, respectively. Similarly, we let K and K_0 denote the number of splits made by the tree construction algorithm and the true number of splits in the tree, so that the true regression tree contains $K_0 + 1$ terminal nodes. Additionally, we vary $n \in \{100, 200, 400, \ldots, 6400\}$. During tree construction, the settings `minsplit=4` and `minbucket=2` are used. The first parameter controls the minimum number of observations required in order to be able to make a split in a terminal node, whereas the second parameter controls the minimum

number of observations that are allowed to be in a terminal node. This choice of parameter values means that splits are never made on nodes with less than four observations and terminal nodes must contain at least two observations. For each simulation sub-setting, 2500 realizations are produced. The simulation study is implemented in R.

4.1 Metrics Used

To assess each pruning algorithm's ability to recover the correct number of terminal nodes in Settings 1 to 3, we consider two metrics. For the following definitions we write $\mathcal{P}'$ to denote a partition that is obtained after using any one of the pruning algorithms. The first metric of interest is $P(|\mathcal{P}'| = K_0 + 1)$, the probability that the pruning algorithm recovers the correct number of terminal nodes. Ideally, this quantity should approach one as the sample size goes to infinity. The second metric is $\mathrm{E}(|\mathcal{P}'|)$, the expected number of terminal nodes in the pruned tree. We will use this quantity in the graphical displays of Chapter 5 because it provides more information about the size of the pruned trees when these do not have the same number of terminal nodes as the true tree. Note that $\mathrm{E}(|\mathcal{P}'|) \to K_0 + 1$ does not necessarily imply that $P(|\mathcal{P}'| = K_0 + 1) \to 1$, and so one should be cautious when making conclusions based on the plots.

It is important to note that if a pruning algorithm does not recover the correct number of terminal nodes, this can indicate an issue with the *construction* algorithm. For example, this can occur if the wrong variable is selected for the first split. In such cases, it is reasonable to expect the pruning algorithms to select trees with a larger number of terminal nodes in order to compensate for the wrong split and to recover the approximate tree structure. A rough check for whether the construction algorithm is on the right track can be made by recording the proportion of simulation realizations in which the variable $X^{(1)}$ was chosen for the first split during construction (the first split should always be on this variable in the simulation settings considered). We use this measure because it is easier to assess than a record of whether all splits were made on the right variables in locations close to the true split-points in the entire tree.

To assess the predictive performance of trees obtained using a given pruning algorithm, a natural measure is the mean squared prediction error (MSPE). For the following definition, we write $m(\mathbf{x})$ to denote the prediction at $\mathbf{x}$ from a tree obtained using one of the pruning algorithms. This pruned tree is random and dependent on the data $(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)$. We let $(\mathbf{X}^*, Y^*)$ be independent of the data but with the same distribution as $(\mathbf{X}, Y)$ and define

$$\mathrm{MSPE} = \mathrm{E}((Y^* - m(\mathbf{X}^*))^2),$$

where the expectation is taken over all of $(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)$, and $(\mathbf{X}^*, Y^*)$. We approximate this quantity conditional on the true regression function (which may change in some settings) and $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ by drawing 2500 independent observations of $(\mathbf{X}^*, Y^*)$

distributed according to $(\mathbf{X}, Y)$ for *each* simulation realization. The estimated MSPE conditional on $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ is then averaged over the 2500 simulation realizations to obtain an estimate of the full expectation. We also consider the relative MSPE (rel-MSPE), which is estimated by taking the estimated MSPE conditional on $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ divided by the smallest such estimated MSPE obtained by all pruning algorithms and averaging the fractions over the 2500 simulation realizations.

4.2 Setting 1: Simple Scenarios

For Setting 1, we consider a variety of simple sub-settings that can be used to assess the performance of the pruning algorithms in well-behaved scenarios. We consider $p = 5$, $p_0 \in \{2, 5\}$, $K \in \{7, 15\}$, $K_0 = 5$, and $\sigma^2 \in \{0.01, 1\}$. In this and other settings, the values of K are always chosen to be one less than a power of two to allow for a fair comparison between our tree construction algorithm (which uses the "best first" approach) and that of `rpart()`, because the latter can be controlled with a parameter, `maxdepth`, that specifies the maximum depth of the tree instead of the number of splits.

The true trees for $p_0 = 2, 5$ are displayed in Figures 4.1 and 4.2, respectively. The means in the terminal nodes are chosen so that (at least) the first split can be easily located when $\sigma^2 = 0.01$, but splitting on the correct first variables is potentially more difficult when $\sigma^2 = 1$. We additionally consider settings in which the mean values in the terminal nodes are randomly permuted for each simulation realization, which we refer to as random trees, in contrast to the fixed trees displayed in the figures. The random trees are important to include in the simulation study because they allow conclusions based on the simulation results to be drawn to a wider range of tree structures. In particular, the random trees include trees that can make it difficult for the tree construction algorithm to recover the true tree structure.

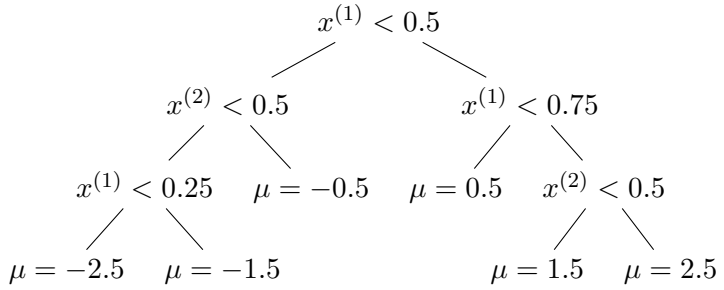

Figure 4.1: True regression tree for Setting 1 with $p_0 = 2$.

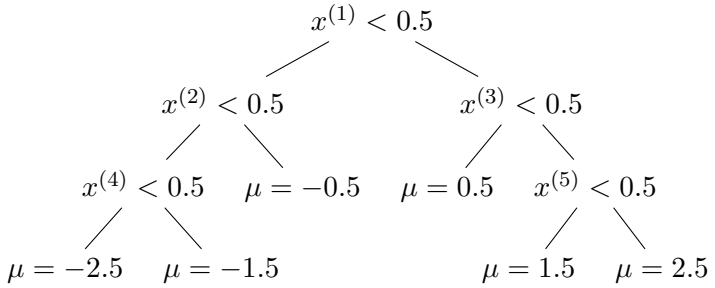

Figure 4.2: True regression tree for Setting 1 with $p_0 = 5$.

4.3 Setting 2: Many Variables and Correlation

Setting 2 considers the effects of correlation on the tree construction and pruning algorithms when only a few of the explanatory variables are related to the response. In the most extreme case with only two variables that are perfectly correlated, it is impossible to know which variable is the "correct" one to split on. Asymptotically, we believe that a correlation between variables that is less than one in magnitude should not cause issues during tree pruning. However, we wish to know whether an effect of correlation on tree construction or pruning is noticeable in finite samples. (Note that Lemmas 1 and 2 and Theorem 1 *allow* for correlated explanatory variables. However, we do require a non-degenerate density function, which implies that the variables are not allowed to be perfectly correlated.)

Each $\mathbf{X}_i$ is drawn from a p-dimensional multivariate normal distribution with mean equal to 0.5 in each dimension. The marginal variance of each component is set equal to 1/16—this results in observed values $\mathbf{x}_i$ that are mostly in $[0, 1]^p$—and the correlation between each pair of components is set equal to $\rho = 0.9$.

A relatively large number of variables is used, with $p = 25$. Only $p_0 = 5$ of these are related to the response. We use $K = 15$ and set $K_0 = 7$, with $\sigma^2 \in \{0.01, 1\}$. The regression tree is given in Figure 4.3. As in Setting 1 with the simple trees, we also try a random version of the tree in which the means in the terminal nodes are randomly permuted for each simulation realization.

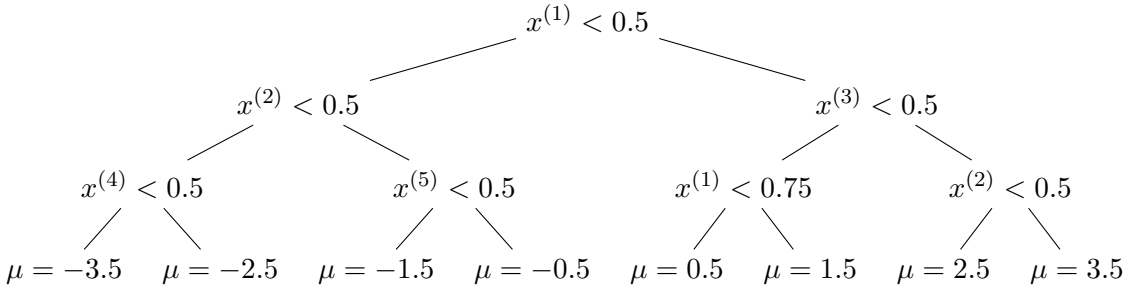

Figure 4.3: True regression tree for Setting 2.

4.4 Setting 3: Deep Trees

Next, we consider deep trees. The BIC-ET and BIC-BF algorithms repeatedly makes decisions whether to merge two nodes. If the fitted tree is much deeper than the true tree, a large sample size might be required until all merge decisions are successfully passed. For example, if 40 unnecessary splits have been made by the regression tree construction algorithm and a merge decision is made correctly 95% of the time, the probability of recovering the true tree exactly is less than 12.2%. In contrast, the BIC-AI and CART pruning algorithms do not face this issue and we expect them to perform quite well.

We set $p = p_0 = 5$, $K = 63$, $K_0 = 19$, and use $\sigma^2 \in \{0.01, 1\}$. The tree is displayed in Figure 4.4. We again use a random version of the tree with permuted terminal node means. We omit the sub-settings with $n = 100$ because it is not possible to construct a tree with 64 terminal nodes and at least two observations in each terminal node in this case.

4.5 Setting 4: True Regression Function Is Not a Tree

Finally, Setting 4 considers the case where the tree underlying regression function *cannot* be represented by a tree. This setting is relevant primarily for the evaluation of the prediction accuracy of the trees obtained with the various pruning algorithms. We set $p = p_0 = 2$ and vary $K \in \{31, 63\}$, with $\sigma^2 \in \{0.01, 1/9\}$. A different choice of values for σ^2 is used in this setting, because the regression function is now

$$m(\mathbf{x}) = \sin(2\pi x^{(1)}) \cdot \sin(2\pi x^{(2)}),$$

a surface with two peaks and two troughs on $[0, 1]^2$, as can be seen in Figure 4.5. With enough data and a large enough regression tree, it is possible to approximate the true surface to an arbitrary level of precision [1]. However, we believe that it will not be easy for a single regression tree to achieve this with only a moderate sample size. As in Setting 3 with deep trees, we omit the sub-settings where both $K = 63$ and $n = 100$.

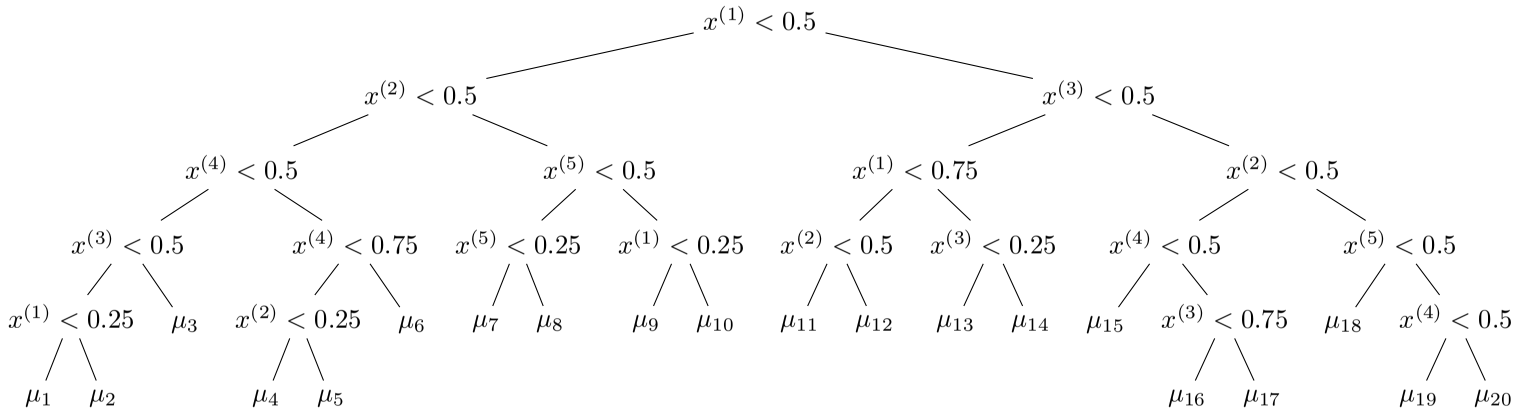

Figure 4.4: True regression tree for Setting 3. For the fixed tree, $\mu_k = -2.5 + 5(k-1)/19$, $k = 1, 2, \ldots, 20$ so that $-2.5 \leq \mu_k \leq 2.5$. In the random tree, these values are permuted randomly for each simulation realization.

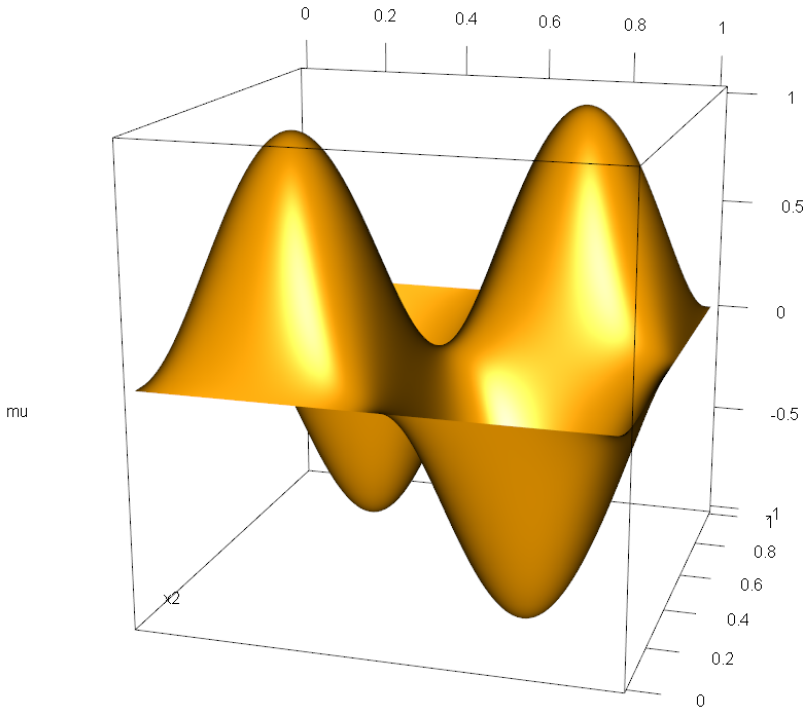

Figure 4.5: The true regression function for Setting 4 restricted to $[0, 1]^2$.

Chapter 5

Simulation Results

In this chapter we present the results of the simulation study that was laid out in Chapter 4. First, we assess each pruning algorithm's ability to recover the true model dimension (number of terminal nodes). Then, we compare the prediction accuracy of the trees obtained with each of the pruning algorithms in terms of their relative mean squared prediction error (rel-MSPE). We compute pairwise differences of the rel-MSPEs between BIC-AI and CART pruning and compare the two-sided p-values from a paired z-test to $0.05/7 \approx 0.0071$ or $0.05/6 \approx 0.0083$ to account for multiple comparisons with different sample sizes in any given sub-setting.

5.1 Model Dimension Recovery

In Setting 1 we consider simple trees. The simulation results are provided in Figure 5.1 for fixed trees and in Figure 5.2 for random trees. Empirically, the expected number of terminal nodes in the trees pruned with BIC-AI converges to the true value of $K_0 + 1 = 6$ in all settings with the fixed true trees (where means in the terminal nodes are fixed). For smaller samples, BIC-AI produces trees with more than six terminal nodes, on average. With the fixed trees, the CART pruning algorithm performs similar to BIC-AI when the sample size is large. For smaller sample sizes, CART pruning favours smaller trees than BIC-AI, on average. Here, BIC-BF is similar in performance to BIC-AI for the smaller value of $K = 7$, but convergence is questionable and considerably slower for BIC-BF when $K = 15$.

In the random tree sub-setting for Setting 1 (where means in the terminal nodes are permuted), the true tree structure seems to be difficult to recover during the tree construction process (the first split was made on the correct variable approximately 42% or 74% of the time) and all pruning algorithms tend to favour larger trees. In all cases in the random setting, CART pruning prefers slightly smaller trees or trees equal in size to those obtained with BIC-AI. The AIC-based pruning algorithms perform quite poorly in terms of recovering the true model dimension in all cases.

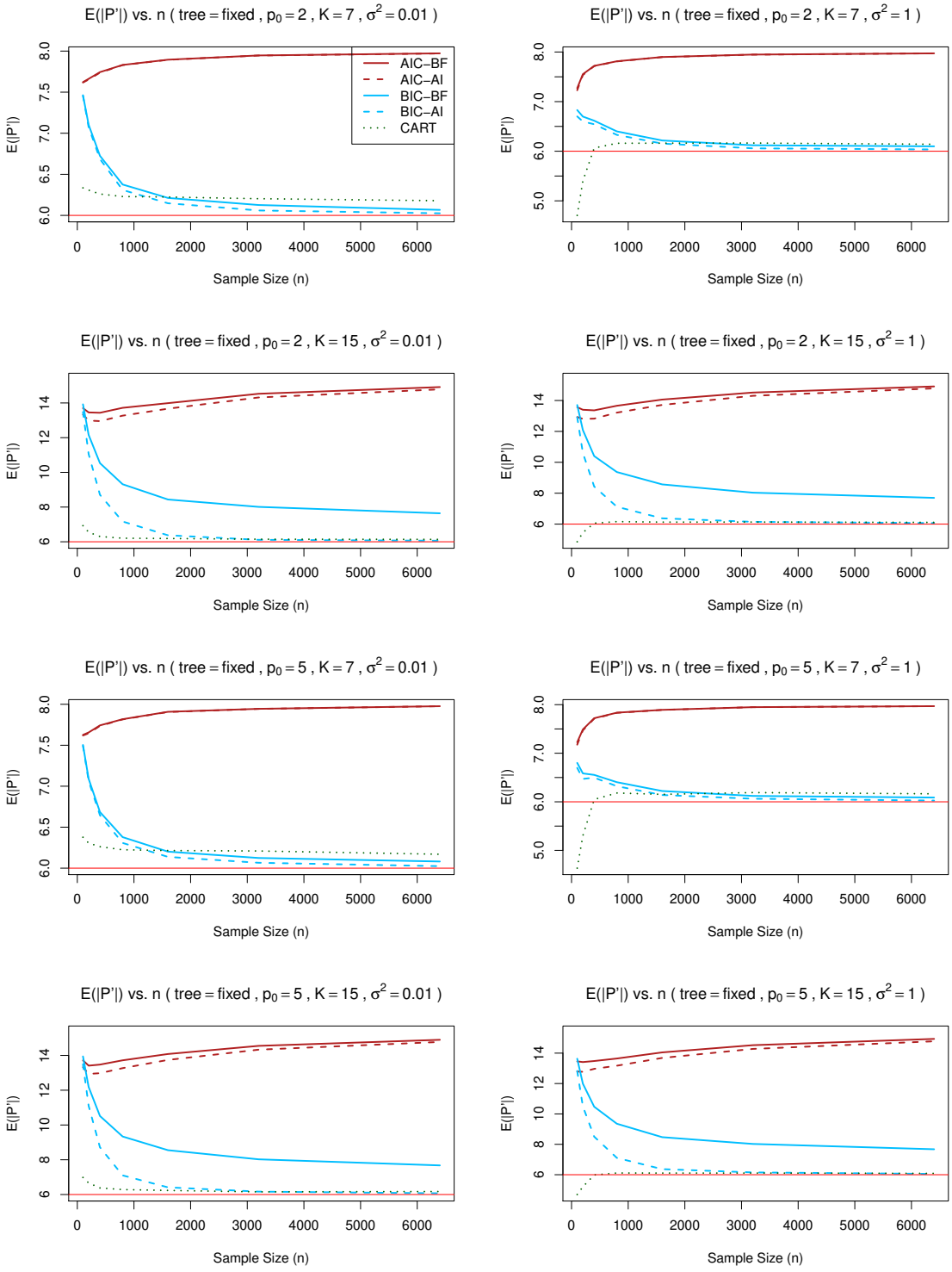

Figure 5.1: Setting 1 results for model dimension recovery (fixed trees). For these settings, $X^{(1)}$ was chosen as the first splitting variable in 100% of the simulation realizations.

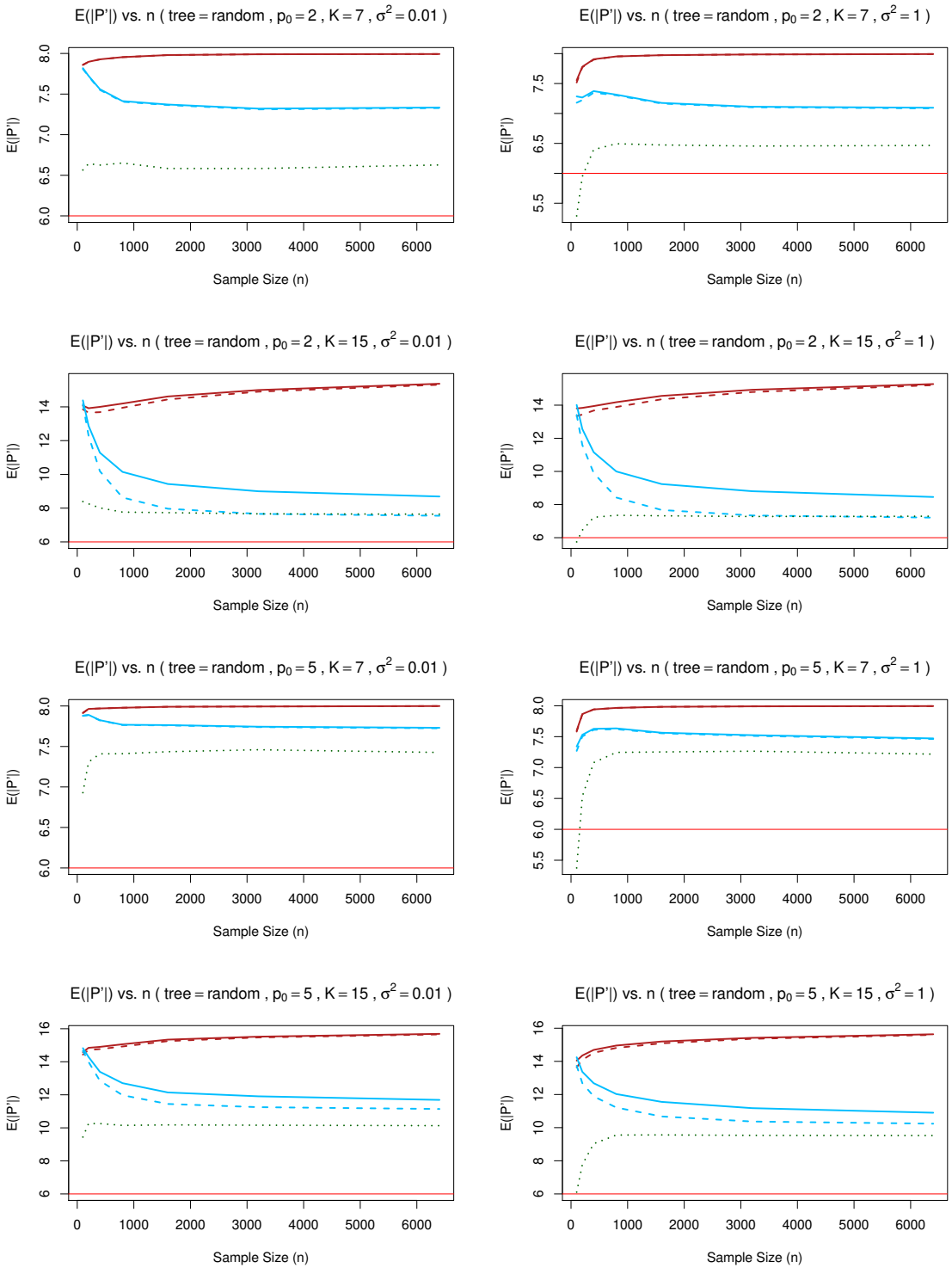

Figure 5.2: Setting 1 results for model dimension recovery (random trees). For the first four settings (reading horizontally), $X^{(1)}$ was chosen as the first splitting variable in approximately 74% of the simulation realizations. In the remaining four settings, $X^{(1)}$ was chosen as the first splitting variable in approximately 42% of the simulation realizations.

The results from Setting 2 with many correlated predictors, few of which are actually related to the response, are displayed in Figure 5.3. With fixed trees, BIC-AI performs quite well but is outperformed by CART pruning in small samples. When we make recovery of the tree structure more difficult with the random trees so that the first splitting variable is chosen correctly approximately 33% of the time, both pruning algorithms select large trees with CART pruning preferring slightly smaller trees. The size of tree that is best for prediction remains to be determined in the next section on prediction accuracy, however.

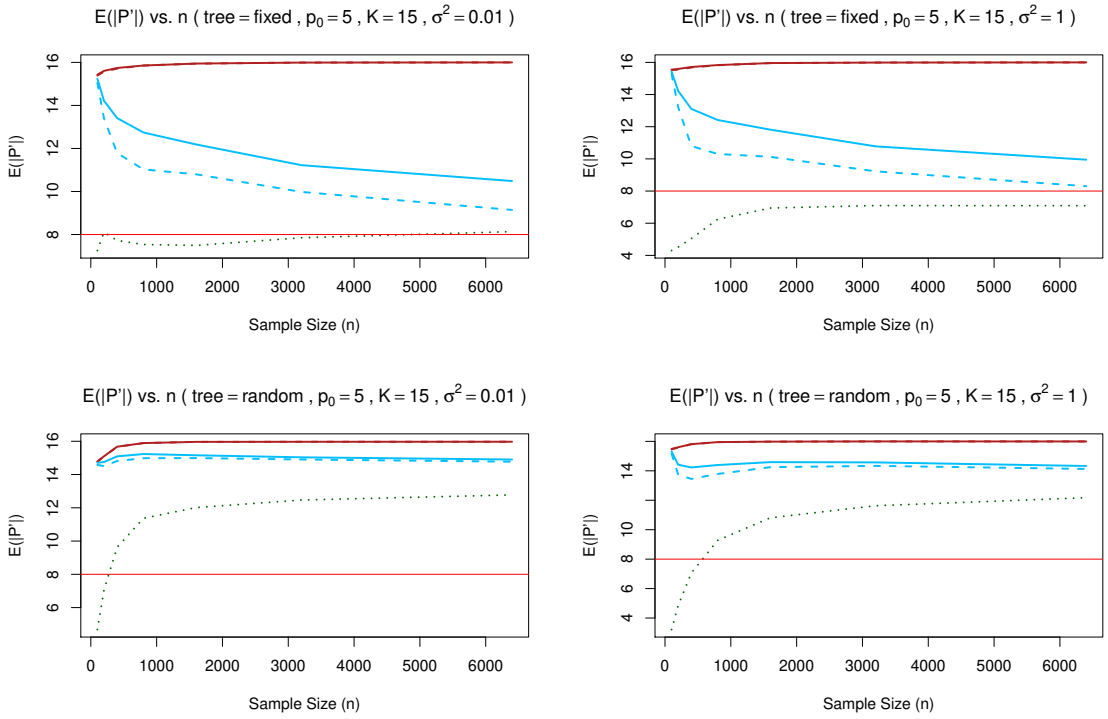

Figure 5.3: Setting 2 results for model dimension recovery. In the first two settings (reading horizontally), $X^{(1)}$ was chosen as the first splitting variable in 100% and 99.8% of the simulation realizations, respectively. In the last two settings, $X^{(1)}$ was chosen as the first splitting variable in approximately 33% of the simulation realizations.

In Setting 3 we fit very deep trees. The model dimension recovery results are presented in Figure 5.4. In the fixed tree setting, BIC-AI performs reasonably well asymptotically. However, for the setting with $\sigma^2 = 0.01$, CART seems to recover the correct number of terminal nodes better than BIC-AI. BIC-BF performs moderately well but with quite slow convergence. In the random setting, BIC-AI favours quite large trees, while CART pruning selects smaller trees. Again, which size of tree is better for prediction in this case remains to be assessed in the next section.

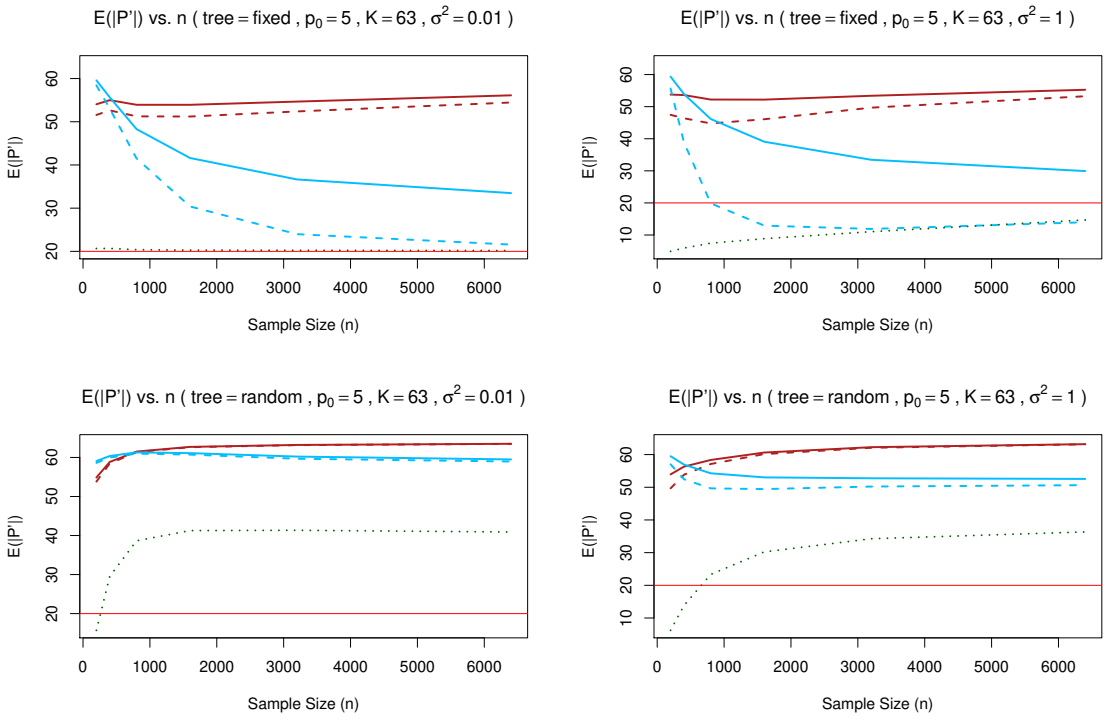

Figure 5.4: Setting 3 results for model dimension recovery. In the first two settings, $X^{(1)}$ was chosen as the first splitting variable in 100% of the simulation realizations. In the last two settings, $X^{(1)}$ was chosen as the first splitting variable in approximately 39% of the simulation realizations.

5.2 Prediction Accuracy

We revisit the simple trees from Setting 1. The plots for prediction accuracy in this setting are provided in Figure 5.5 (fixed trees) and Figure 5.6 (random trees). In all sub-settings with fixed trees, BIC-AI is on par with CART pruning in terms of rel-MSPE when the sample size is sufficiently large. For small sample sizes, CART seems to perform better in terms of rel-MSPE. For example, the relatively small difference in rel-MSPEs between CART pruning and BIC-AI with $p_0 = 2$, $K = 7$, $\sigma^2 = 0.01$, and $n = 800$ is highly significant with a two-sided p-value based on the paired z-test that is less than 1×10^{-5} (a description of the testing procedure is given at the beginning of Chapter 5). However, BIC-AI has an estimated rel-MSPE less than or equal to 1.01 around $n = 1000$, if not earlier. BIC-BF also exhibits fairly good performance in the settings with the fixed trees.

With random trees in Setting 1, BIC-AI can drastically outperform CART pruning, particularly when σ^2 is small. With $p_0 = 2$, $K = 7$, $\sigma^2 = 0.01$, and $n = 6400$, the average rel-MSPE of trees obtained with CART pruning was around 8, which is a remarkably poor score (two-sided p-values comparing BIC-AI and CART rel-MSPEs are less than 3×10^{-16} for all values of n). In the two settings with $K = 15$ and $\sigma^2 = 1$, CART had acceptable or favourable performance. The AIC pruning algorithms have relatively poor MSPE performance but can outperform or perform as well as CART pruning in the random trees setting.

The findings for Setting 2 with correlated variables are similar to the results from Setting 1. The Setting 2 results for prediction accuracy are displayed in Figure 5.7.

The prediction accuracy results with deep trees in Setting 3 are displayed in Figure 5.8. With fixed trees, CART pruning outperforms BIC-AI for small and moderate sample sizes, but the relative MSPE of BIC-AI decreases to around one for large sample sizes (approximately $n = 6400$ and $n = 3200$ for $\sigma^2 = 0.01$ and $\sigma^2 = 1$, respectively). With the fixed trees, all pairwise comparisons are significant with p-values less than 3×10^{-16}. On the other hand, in the random tree setting, CART pruning performs incredibly poorly for large sample sizes and small σ^2 (all 12 two-sided p-values for random trees are less than 3×10^{-16}). This is similar to what we observed in Setting 1. We offer an explanation of this phenomenon in Chapter 6.

Finally, we assess the performance of the pruned regression trees when the true regression function is not a tree. The results are presented in Figure 5.9. We see that the information-based pruning algorithms outperform CART pruning provided that the sample size is sufficiently large (around $n = 800$ or earlier). All 26 two-sided comparisons in rel-MSPEs between CART pruning and BIC-AI are significant with all p-values being less than 2.2×10^{-5}. CART pruning seems to be inadequate in this case, but a more in-depth study is necessary to properly assess its performance in such settings. Interestingly, AIC pruning seems to be better than BIC pruning in this setting.

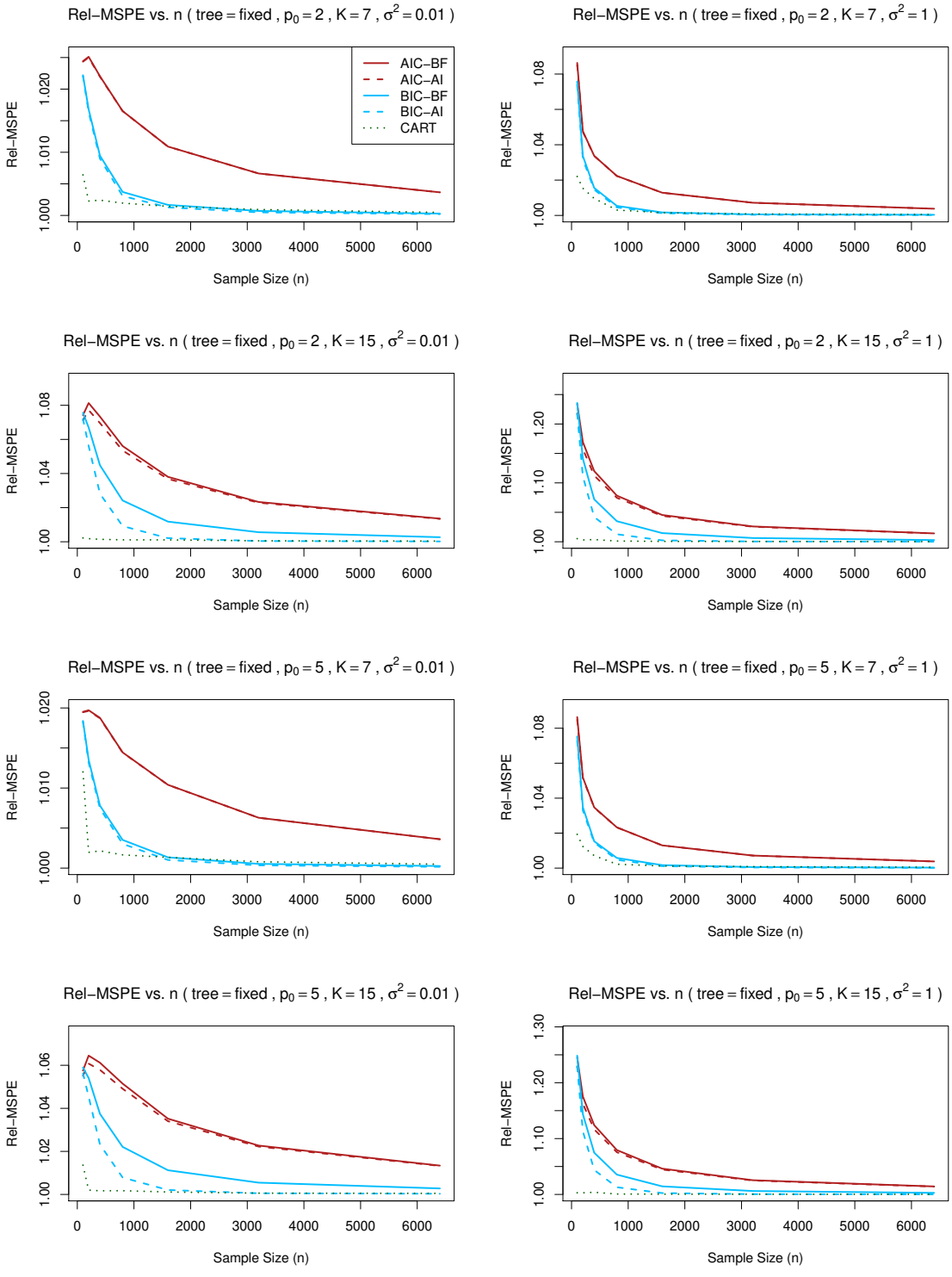

Figure 5.5: Setting 1 results for prediction accuracy (fixed trees).

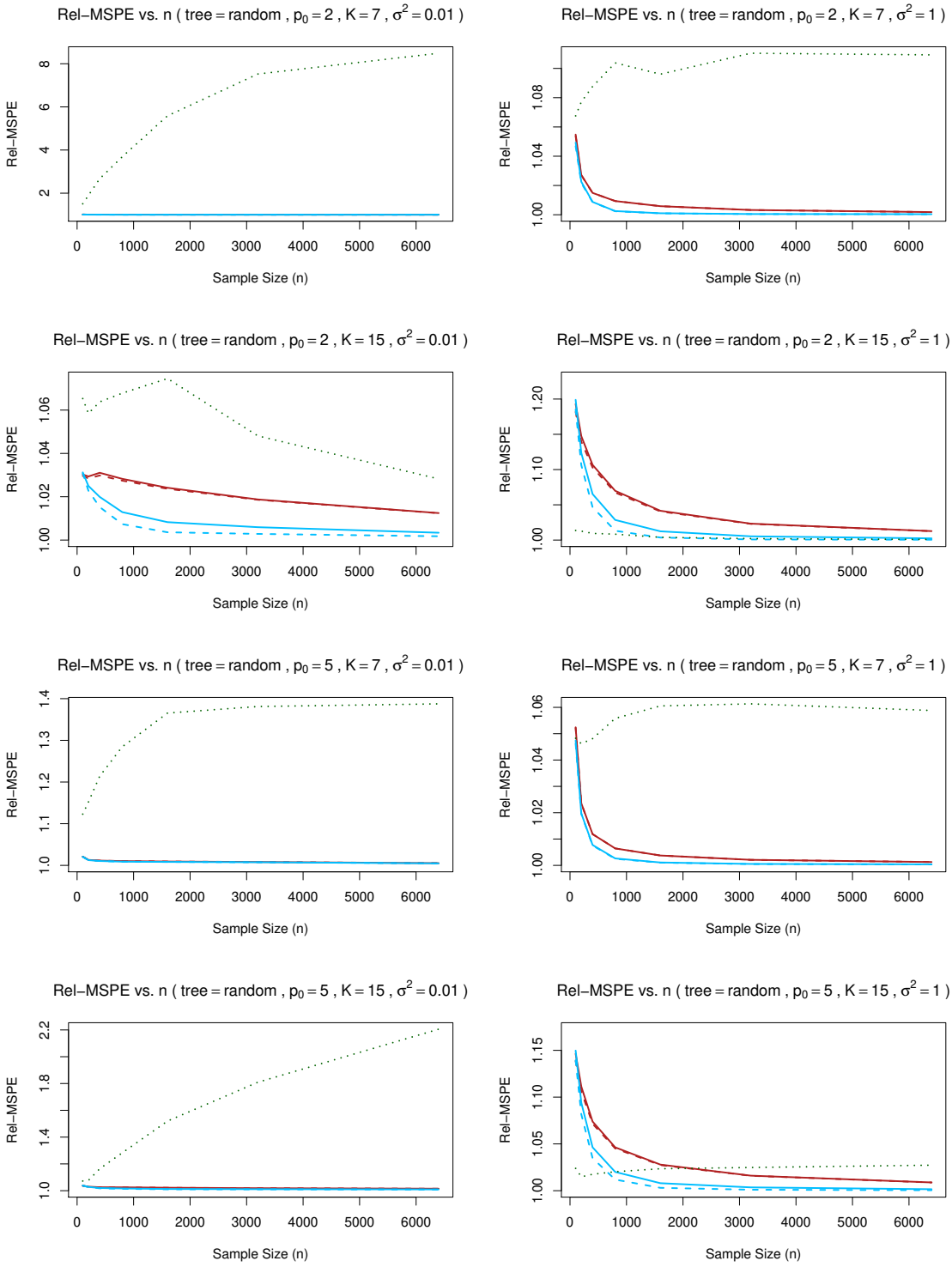

Figure 5.6: Setting 1 results for prediction accuracy (random trees).

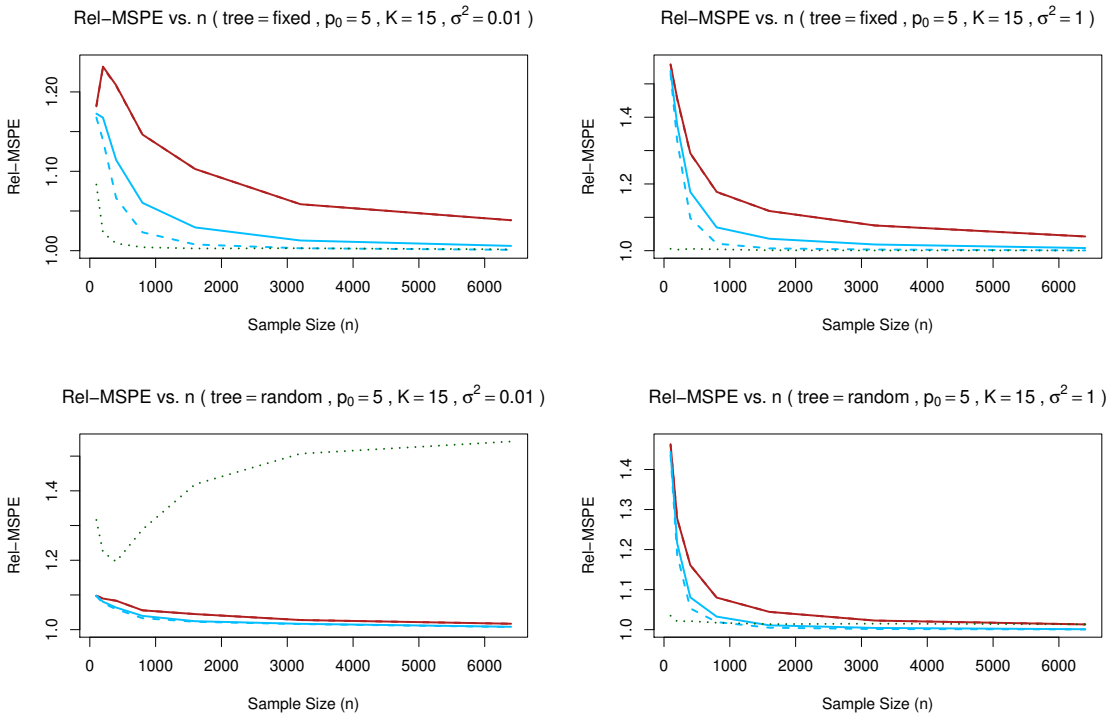

Figure 5.7: Setting 2 results for prediction accuracy.

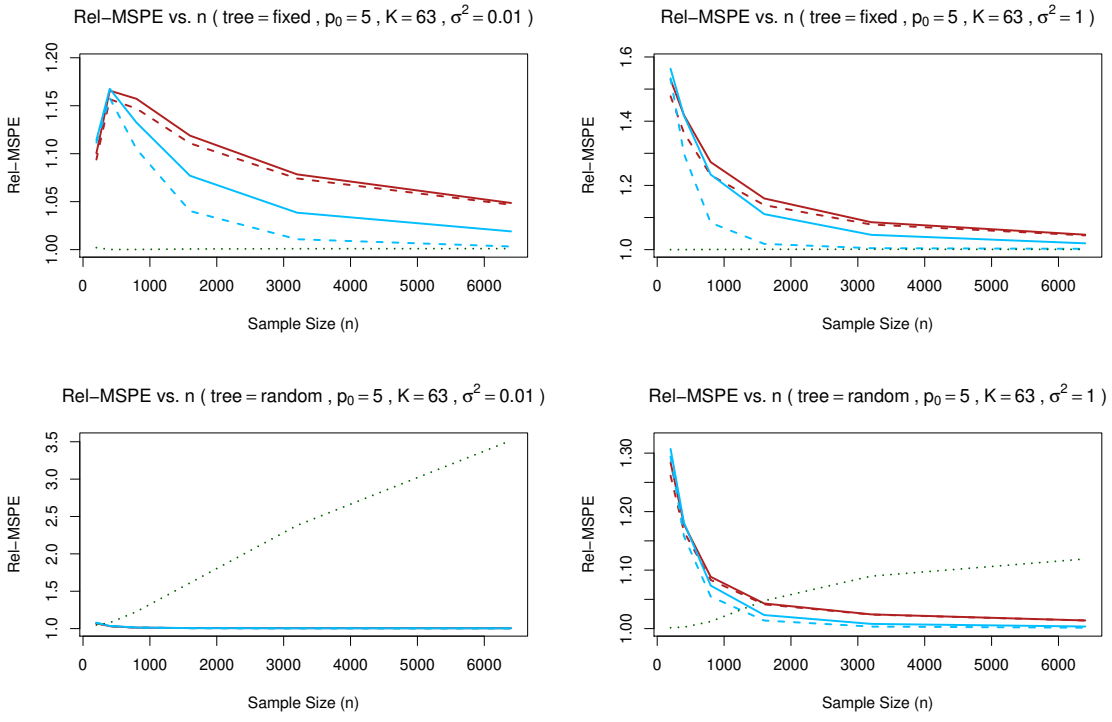

Figure 5.8: Setting 3 results for prediction accuracy.

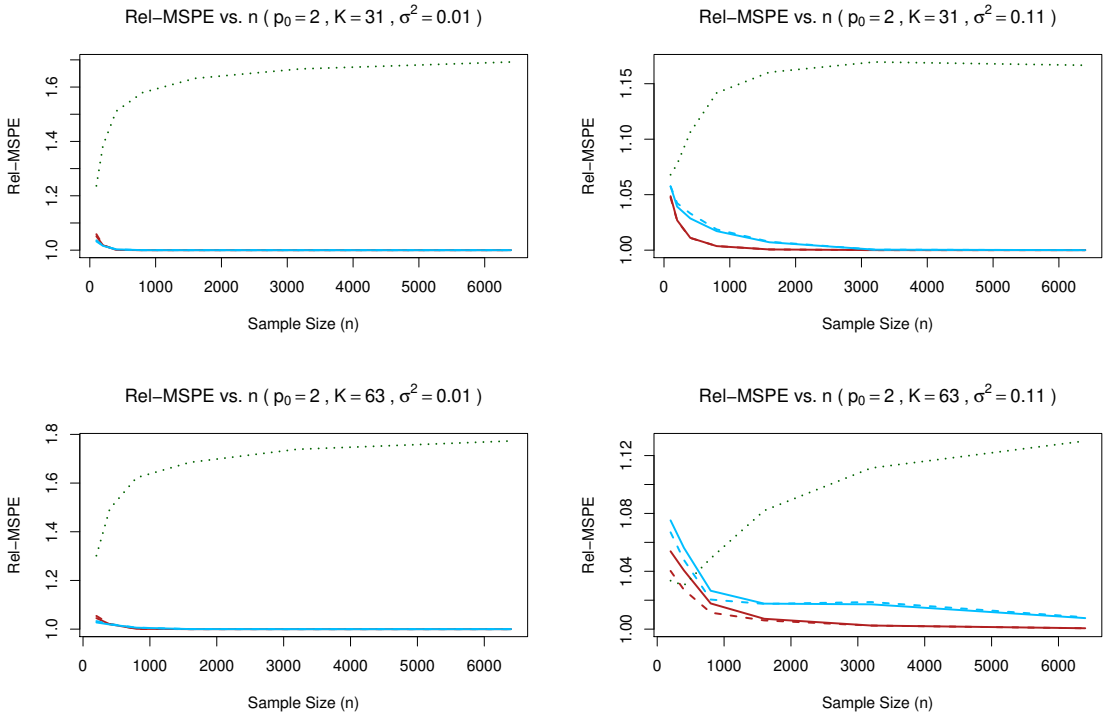

Figure 5.9: Setting 4 results for prediction accuracy.

Chapter 6

Discussion

In this work, we developed and presented several regression tree pruning algorithms that do not rely on cross-validation and can serve as effective alternatives to the standard CART cost-complexity pruning algorithm. In particular, one of these—the BIC-AI (accumulated information) pruning algorithm—proved to be an efficient alternative that produces pruned trees that can have comparable (and sometimes favourable) prediction accuracy in some of the considered settings when compared to standard CART pruning.

We built upon the work of previous authors and also introduced some new developments. Using an existing modified BIC for change-point detection [24], we proposed a slight generalization of this BIC for regression tree pruning and proved its consistency in this new setting. We also incorporated this modified BIC into the accumulated information pruning algorithm of [9] to create the BIC-AI pruning algorithm. Along the way, we introduced the simpler pruning algorithms BIC-BF (branch freezing) and BIC-ET (early termination) that make use of the modified BIC and provided a consistency result for BIC-ET. We also proposed AIC-based pruning algorithms by using another change-point detection result from [16] and [17].

In the "random trees" simulation results of Chapter 5, we saw that trees pruned with BIC-AI can outperform trees obtained with CART pruning by a considerable margin in terms of relative MSPE when the sample size is sufficiently large and σ^2 is small. However, when the sample size is small and the true regression function can be represented by a tree, we found that CART pruning outperforms BIC-AI in the "fixed tree" settings. We also saw with the fixed trees that the expected number of terminal nodes in trees pruned with BIC-AI is asymptotically close to the true number of terminal nodes when the true regression function can be represented by a tree and the tree construction algorithm begins by splitting on the correct variable.

The poor relative MSPE performance of trees obtained with standard CART pruning in settings with random trees and small σ^2 ($= 0.01$) is an unexpected finding of the simulation study. In these cases, pruning approaches such as the proposed BIC-based algorithms should be able to make very good pruning decisions because the noise is small relative to the signal

and pruning decisions can be made at a local level. In contrast, CART pruning chooses an optimal cost-complexity parameter that is applied directly to an entire tree. It is possible for the impact of a true pattern that is captured by a terminal node to be diluted by noise present in terminal nodes belonging to the same branch, resulting in an over-pruned tree with CART pruning.

There are several directions for future research. It would be favourable to generalize the BIC-ET consistency proof to show consistency of the BIC-BF pruning algorithm under weaker assumptions on the tree construction algorithm. In general, the proofs provided in this thesis should be considered as sketches and a work in progress. As was mentioned previously, Theorem 1 relies on a small generalization of Lemma 2 that remains to be proven. Therefore, some more work is needed to firmly establish the theoretical results.

The proposed information-based pruning algorithms can easily be incorporated into the α-pruning algorithm of [9] for random forests. This algorithm uses an information criterion to prune individual trees and obtains forests of slightly pruned trees. A tuning parameter, α, is introduced to control the extent to which the trees are pruned. These pruned trees offer predictions with decreased variability but should not introduce a considerable amount of bias. While [9] used simulated penalty tables to estimate the appropriate information value to attach to nodes or branches of a tree, these values can now be calculated easily with the closed form of our modified BIC. In future work, we plan to examine the performance of BIC-AI with α-pruning in order to tune tree-based ensembles such as random forests.

Bibliography

[1] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and Regression Trees*. CRC Press, 1984.

[2] Antonio Ciampi. Generalized regression trees. *Computational Statistics & Data Analysis*, 12(1):57–78, 1991.

[3] Gerda Claeskens and Nils Lid Hjort. The focused information criterion. *Journal of the American Statistical Association*, 98(464):900–916, 2003.

[4] Jan deLeeuw. Introduction to Akaike (1973) information theory and an extension of the maximum likelihood principle. In *Breakthroughs in Statistics*, pages 599–609. Springer, 1992.

[5] Floriana Esposito, Donato Malerba, and Giovanni Semeraro. A comparative analysis of methods for pruning decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):476–491, 1997.

[6] Floriana Esposito, Donato Malerba, Giovanni Semeraro, and Valentina Tamma. The effects of pruning methods on the predictive accuracy of induced decision trees. *Applied Stochastic Models in Business and Industry*, 15(4):277–299, 1999.

[7] Guangzhe Fan and J Brian Gray. Regression tree analysis using TARGET. *Journal of Computational and Graphical Statistics*, 14(1):206–218, 2005.

[8] Alexis Hannart and Philippe Naveau. An improved Bayesian information criterion for multiple change-point models. *Technometrics*, 54(3):256–268, 2012.

[9] Andrew James Dennis Henrey. *Statistical Learning Tools for Heteroskedastic Data*. PhD thesis, Science: Department of Statistics and Actuarial Science, 2016.

[10] Aram Karalic and Bojan Cestnik. The Bayesian approach to tree-structured regression. In *Proceedings of ITI*, volume 91, pages 155–160, 1991.

[11] Sadanori Konishi and Genshiro Kitagawa. *Information criteria and statistical modeling*. Springer Science & Business Media, 2008.

[12] Colin H LaMont and Paul A Wiggins. The development of an information criterion for change-point analysis. *Neural Computation*, 28(3):594–612, 2016.

[13] Michael LeBlanc and Robert Tibshirani. Monotone shrinkage of trees. *Journal of Computational and Graphical Statistics*, 7(4):417–433, 1998.

[14] Berent Ånund Strømnes Lunde, Tore Selland Kleppe, and Hans Julius Skaug. An information criterion for automatic gradient tree boosting. *arXiv preprint arXiv:2008.05926*, 2020.

[15] John Mingers. An empirical comparison of pruning methods for decision tree induction. *Machine Learning*, 4(2):227–243, 1989.

[16] Yoshiyuki Ninomiya. Information criterion for Gaussian change-point model. *Statistics & Probability Letters*, 72(3):237–247, 2005.

[17] Yoshiyuki Ninomiya. Change-point model selection via AIC. *Annals of the Institute of Statistical Mathematics*, 67(5):943–961, 2015.

[18] John R Quinlan. Learning with continuous classes. In *5th Australian Joint Conference on Artificial Intelligence*, volume 92, pages 343–348. World Scientific, 1992.

[19] Marko Robnik-Šikonja. CORE-a system that predicts continuous variables. In *Proceedings of ERK'97*, pages B145–148, 1997.

[20] Marko Robnik-Šikonja and Igor Kononenko. Pruning regression trees with MDL. In *Proceedings of the 13th European Conference on Artificial Intelligence, John Wiley & Sons, Chichester, England*, pages 455–459, 1998.

[21] Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[22] Erwan Scornet, Gérard Biau, and Jean-Philippe Vert. Consistency of random forests. *The Annals of Statistics*, 43(4):1716–1741, 2015.

[23] Shayle R Searle and Marvin HJ Gruber. *Linear Models*. John Wiley & Sons, Incorporated, 2016.

[24] Gang Shen and Jayanta K Ghosh. Developing a new BIC for detecting change-points. *Journal of Statistical Planning and Inference*, 141(4):1436–1447, 2011.

[25] David J Spiegelhalter, Nicola G Best, Bradley P Carlin, and Angelika Van Der Linde. Bayesian measures of model complexity and fit. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 64(4):583–639, 2002.

[26] David J Spiegelhalter, Nicola G Best, Bradley P Carlin, and Angelika Van der Linde. The deviance information criterion: 12 years on. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, pages 485–493, 2014.

[27] Xiaogang Su, Morgan Wang, and Juanjuan Fan. Maximum likelihood regression trees. *Journal of Computational and Graphical Statistics*, 13(3):586–598, 2004.

[28] Luís Fernando Raínho Alves Torgo. *Inductive learning of tree-based regression models*. PhD thesis, Universidade do Porto. Reitoria, 1999.

[29] Aad W Van der Vaart. *Asymptotic Statistics*. Cambridge University Press, 2000.

[30] Sumio Watanabe. A widely applicable Bayesian information criterion. *Journal of Machine Learning Research*, 14(Mar):867–897, 2013.

[31] Yi-Ching Yao. Estimating the number of change-points via Schwarz' criterion. *Statistics & Probability Letters*, 6(3):181–189, 1988.

Appendix A

Proofs

In this appendix, we lay out the proofs of four lemmas and a theorem. If not provided here, the lemma/theorem statements can be found in Section 3.5.1. Lemma 1 states the consistency of the estimators of the splitting variables and split locations in the case of a "tree stump". It is independent of the three other lemmas and the theorem. Lemma 2 shows that the modified BIC consistently selects either the "no split" or "single split" (tree stump) model, when one of these is the correct model. Theorem 1 then shows that BIC-ET recovers the true subtree under appropriate conditions. Lemma 2 and Lemma 3 are important tools used in the proof of the theorem. Lemma 4 shows that one can replace an unknown σ^2 with a consistent estimate and still have consistency of the $lBIC$. As was mentioned in Chapter 6, the proofs provided in this thesis should be considered as sketches and a work in progress.

A.1 Lemma 1

Lemma 1 is independent of the other lemmas and the theorem. It is only used to justify some of the conditions from Section 3.5.1 and plays no role in the consistency of the $lBIC$.

Proof of Lemma 1. We make use of Theorem 5.7 from [29] and use the same notation and proof approach that is presented there. Without loss of generality, assume $j_0 = 1$. Clearly, Θ is compact. Fix $\theta \in \Theta$ and let

$$f((\mathbf{x}, y), \theta) = -\left(y - \mu_1 \mathbb{1}(x^{(j)} < z) - \mu_2 \mathbb{1}(x^{(j)} \geq z)\right)^2.$$

Further, let $M(\theta) = \mathrm{E}(f((\mathbf{X}, Y), \theta))$.

We first show that

$$\sup_{\theta \in \Theta} |M_n(\theta) - M(\theta)| \xrightarrow{as*} 0.$$

To this end, let $\mathcal{F} = \{f((\mathbf{x}, y), \theta) : \theta \in \Theta\}$. We simply have to show that $\mathcal{F}$ is P-Glivenko-Cantelli. The argument is essentially the same as for Example 19.8 of [29], with a very slight

weakening of the continuity assumption. Note that

$$|f((\mathbf{x}, y), \theta)| \leq (|y| + |\mu_1| + |\mu_2|)^2 \leq (|y| + 2K_\mu)^2 \,,$$

so $\mathcal{F}$ has an integrable envelope function, $F((\mathbf{x}, y)) = (|y| + 2K_\mu)^2$. Further, observe that for any $\theta^* \in \Theta$, the map $\theta \mapsto f((\mathbf{x}, y), \theta)$ is continuous at θ^* for all $(\mathbf{x}, y)$ provided that $x^{(j^*)} \neq z^*$. Because $\mathbf{X}$ has positive density with respect to Lebesgue measure on $[0, 1]^p$, this set has probability zero. (The main point is that the exceptional set of $\mathbf{x}$ is allowed to depend on θ^*.) Verifying each of the steps in the mentioned example, we have the desired uniform convergence.

Next, we show that for all $\epsilon > 0$,

$$\sup_{\theta : \|\theta - \theta_0\| \geq \epsilon} M(\theta) < M(\theta_0),$$

where $\|\theta - \theta_0\|$ is the Euclidean distance between the vectors θ and θ_0. First, some work shows that

$$-M(\theta) = \begin{cases} -M_1(\theta), & j = j_0, z = z_0 \\ -M_2(\theta), & j = j_0, z < z_0 \\ -M_3(\theta), & j = j_0, z > z_0 \\ -M_4(\theta), & j \neq j_0, \end{cases}$$

where

$$\begin{aligned}
-M_1(\theta) &= P(X^{(j_0)} < z_0)(\mu_1 - \mu_{1,0})^2 + P(X^{(j_0)} \geq z_0)(\mu_2 - \mu_{2,0})^2 + \sigma^2, \\
-M_2(\theta) &= P(X^{(j_0)} < z)(\mu_1 - \mu_{1,0})^2 + P(z \leq X^{(j_0)} < z_0)(\mu_2 - \mu_{1,0})^2 \\
&\quad + P(X^{(j_0)} \geq z_0)(\mu_2 - \mu_{2,0})^2 + \sigma^2, \\
-M_3(\theta) &= P(X^{(j_0)} < z_0)(\mu_1 - \mu_{1,0})^2 + P(z_0 \leq X^{(j_0)} < z)(\mu_1 - \mu_{2,0})^2 \\
&\quad + P(X^{(j_0)} \geq z)(\mu_2 - \mu_{2,0})^2 + \sigma^2, \\
-M_4(\theta) &= P(X^{(j)} < z, X^{(j_0)} < z_0)(\mu_1 - \mu_{1,0})^2 + P(X^{(j)} < z, X^{(j_0)} \geq z_0)(\mu_1 - \mu_{2,0})^2 \\
&\quad + P(X^{(j)} \geq z, X^{(j_0)} < z_0)(\mu_2 - \mu_{1,0})^2 + P(X^{(j)} \geq z, X^{(j_0)} \geq z_0)(\mu_2 - \mu_{2,0})^2 + \sigma^2.
\end{aligned}$$

Clearly, $M(\theta) \leq -\sigma^2$ for any θ and $M(\theta_0) = -\sigma^2$. Let $\epsilon > 0$ and consider θ with $\|\theta - \theta_0\| \geq \epsilon$. Some tedious work (after checking finitely many cases) shows that the supremum of $M(\theta)$ over these θ is strictly less than $M(\theta_0)$ by (A1), (B1), and (B2).

Finally, by definition, we have $M_n(\hat{\theta}_n) \geq M_n(\theta_0)$, and therefore $\hat{\theta}_n \xrightarrow{p} \theta_0$. This implies that each of the components converge in probability to their true values and that $P(j_n = j_0) \to 1$, because j takes on finitely many values. □

A.2 Lemma 2

Proof of Lemma 2. We split the proof into the two cases. This proof is for the version of the $lBIC$ where σ^2 is known. For the (more realistic) case where σ^2 is replaced with a consistent estimator $\hat{\sigma}_n^2$, see Lemma 4. For properties of quadratic forms, see [23].

Case 1
We define a few quantities. The notation is presented in Section 3.2. Let

$$V_{n_L,j} = \sum_{i=1}^{n}(Y_i - \bar{Y})^2 - \sum_{i=1}^{n_L}(Y_i^{*j} - \bar{Y}_{L,n_L}^{*j})^2 - \sum_{i=n_L+1}^{n}(Y_i^{*j} - \bar{Y}_{R,n_L}^{*j})^2,$$

$$S = \left\{ 3\log(n) \leq \max_{\substack{n_L=1,\dots,n-1 \\ j=1,\dots,p}} \frac{1}{\sigma^2} V_{n_L,j} \right\},$$

$$S_{n_L,j} = \left\{ 3\log(n) \leq \frac{1}{\sigma^2} V_{n_L,j} \right\}.$$

We know

$$\begin{aligned} P(lBIC_{0,n}([0,1]^p) \geq lBIC_{1,n}([0,1]^p)) &= P(S) \\ &= P\left(\bigcup_{\substack{n_L=1,\dots,n-1 \\ j=1,\dots,p}} S_{n_L,j} \right) \\ &\leq \sum_{\substack{n_L=1,\dots,n-1 \\ j=1,\dots,p}} P(S_{n_L,j}). \end{aligned}$$

Now, if $Z \sim N(0,1)$,

$$\begin{aligned} P(S_{n_L,j}) &= P(Z^2 \geq 3\log(n)) \\ &= P\left(|Z| \geq \sqrt{3\log(n)} \right) \\ &\leq \exp(-3/2\log(n))) \\ &= n^{-3/2}. \end{aligned}$$

This implies that

$$P(lBIC_{1,n}([0,1]^p) > lBIC_{0,n}([0,1]^p)) \geq 1 - \frac{(n-1)p}{n^{3/2}} \to 1,$$

as $n \to \infty$ for fixed p. This completes the proof for Case 1.

Case 2
We introduce some notation for this case. Let

$$N_L = \sum_{i=1}^{n} \mathbb{1}(X_i^{(j_0)} < z_0),$$

the number of observations to the "left" of z_0 along the j_0-th variable (a random quantity). Define

$$\bar{Y}_{L,z_0} = \frac{\sum_{i=1}^n Y_i \mathbb{1}(X_i^{(j_0)} < z_0)}{N_L},$$

the average of the responses to the "left". We define $\bar{Y}_{R,z_0}$ analogously, but the inequality in the indicator function is not strict in the latter case. Now, let

$$D = \sum_{i=1}^n (Y_i - \bar{Y})^2 - \sum_{i=1}^n (Y_i - \bar{Y}_{L,z_0})^2 \mathbb{1}(X_i^{(j_0)} < z_0) - \sum_{i=1}^n (Y_i - \bar{Y}_{R,z_0})^2 \mathbb{1}(X_i^{(j_0)} \geq z_0).$$

Then,

$$P(lBIC_{0,n}([0,1]^p) > lBIC_{1,n}([0,1]^p)) \geq P(\{1/\sigma^2 D > 3\log(n)\} \cap \{N_L \notin \{0,n\}\}).$$

Clearly, $P(N_L \notin \{0,n\}) \to 1$ by (A1) and (B2), so we need to show that $P(1/\sigma^2 D > 3\log(n)) \to 1$. For convenience, we examine the reverse inequality. Let

$$G = \left\{N_L < \sqrt{n}\right\} \cup \left\{N_L > n - \sqrt{n}\right\}.$$

Then,

$$P(1/\sigma^2 D \leq 3\log(n)) = P\left(\{1/\sigma^2 D \leq 3\log(n)\} \cap G\right) + P\left(\{1/\sigma^2 D \leq 3\log(n)\} \cap G^c\right). \tag{A.1}$$

We first show that the left term in (A.1) goes to zero, and then later show the same for the right term, which requires a bit more work. Note that because $\mathbf{X}$ has strictly positive density on $[0,1]^p$, and because $z_0 \in (0,1)$, by the weak law of large numbers,

$$P\left(\left|\frac{N_L}{n} - c\right| > \epsilon\right) \to 0,$$

for any $\epsilon > 0$, where $c = P(X_1^{(j_0)} < z_0) > 0$. This immediately implies that $P(N_L/n > c + \epsilon) \to 0$ and $P(N_L/n < c - \epsilon) \to 0$ for all $\epsilon > 0$. Thus,

$$P(N_L \leq \sqrt{n}) = P(N_L/n \leq 1/\sqrt{n}) \to 0.$$

Similarly,

$$P(N_L \geq n - \sqrt{n}) = P(N_L/n \geq 1 - 1/\sqrt{n}) \to 0.$$

Therefore, $P(G) \to 0$. We now only have to show that the right term in (A.1) goes to zero. To this end, observe that for $n_L \in \{1, \ldots, n-1\}$,

$$P(1/\sigma^2 D \leq 3\log(n)) | N_L = n_L) = P(Q_{\lambda(n_L)} \leq 3\log(n)),$$

where $Q \sim \chi_1^2(\lambda(n_L))$, a non-central chi-square distribution with one degree of freedom and non-centrality parameter

$$\lambda(n_L) = \frac{n_L(n-n_L)}{2n\sigma^2}(\mu_{1,0} - \mu_{2,0})^2,$$

using the parameterization presented in [23]. Note that for $n_L \in [\sqrt{n}, n-\sqrt{n}]$, the minimum of $\lambda(n_L)$ is

$$\lambda_{min} = \lambda(\sqrt{n}) = \lambda(n-\sqrt{n}) = \frac{\sqrt{n}(n-\sqrt{n})}{2n\sigma^2}(\mu_{1,0}-\mu_{2,0})^2.$$

Therefore, for any sufficiently small $t > 0$, using the moment generating function of the non-central chi-square distribution (see [23]),

$$\begin{aligned}
&P(\{1/\sigma^2 D \le 3\log(n)\} \cap G^c) \\
&\quad = P(\{1/\sigma^2 D \le 3\log(n)\} \cap \{N_L = \lceil \sqrt{n} \rceil\}) \\
&\qquad + \ldots + P(\{1/\sigma^2 D \le 3\log(n)\} \cap \{N_L = \lfloor n-\sqrt{n} \rfloor\}) \\
&\quad \le P\left(Q_{\lambda_{min}} \le 3\log(n)\right) \\
&\quad = P(-tQ_{\lambda_{min}} \ge -3t\log(n)) \\
&\quad = P(\exp(-tQ_{\lambda_{min}}) \ge n^{-3t}) \\
&\quad \le \mathrm{E}(\exp(-tQ_{\lambda_{min}})) \cdot n^{3t} \\
&\quad = \frac{1}{(1+2t)^{1/2}} \exp\left\{ -\frac{\sqrt{n}(n-\sqrt{n})}{2n\sigma^2}(\mu_{1,0}-\mu_{2,0})^2[1-(1+2t)^{-1}] + 3t\log(n) \right\} \\
&\quad = \frac{1}{(1+2t)^{1/2}} \exp\left\{ -\frac{1}{2\sigma^2}\sqrt{n}\left(1-\frac{1}{\sqrt{n}}\right)(\mu_{1,0}-\mu_{2,0})^2[1-(1+2t)^{-1}] + 3t\log(n) \right\} \\
&\quad \to 0,
\end{aligned}$$

as $n \to \infty$, because $(\mu_{1,0}-\mu_{2,0})^2$ and $1-1/(1+2t)$ are both strictly greater than zero. This completes the proof for Case 2. □

A.3 Theorem 1

In this section we prove the main result, presented in Theorem 1: BIC-ET recovers the true tree structure under appropriate conditions. Before this, we introduce a somewhat trivial lemma that is used in the proof.

Lemma 3. *Consider sequences of events A_n, B_n such that $P(A_n) \to 1$. Then,*

$$P(B_n) - P(A_n \cap B_n) \to 0.$$

Proof.

$$\begin{aligned} |P(B_n) - P(A_n \cap B_n)| &= P(A_n^c \cap B_n) \\ &\leq P(A_n^c) \\ &\to 0. \end{aligned}$$

□

We can now prove the main theorem (for which we assume σ^2 is known). For the extension to unknown σ^2 replaced with a consistent estimator $\hat{\sigma}_n^2$, see Lemma 4.

Proof of Theorem 1. With the above assumptions, there are only *finitely* many possibilities for the region $\text{Snap}(A_{K-k,n}(\tilde{\epsilon}), \mathcal{G}(\tilde{\epsilon}/N))$ across all n, for any given $1 \leq k \leq K$. This is because $\mathcal{G}(\tilde{\epsilon}/N)$ contains finitely many points and $\mathbf{X}$ has support on $[0,1]^p$.

With this in mind, let

$$C_{K-k,n} = \text{Snap}(A_{K-k,n}(\tilde{\epsilon}), \mathcal{G}(\tilde{\epsilon}/N))$$

for each k and n. The region $C_{K-k,n}$ is random. As was just mentioned, the set of possible values of $C_{K-k,n}$ across all k, n is finite. Let $\mathcal{C}_{K-k}$ be this finite set of possible values across n for a fixed value of k. For a rectangular, axis-aligned, and closed region $A \subset [0,1]^p$, using the definitions given by equations (3.1) and (3.3), let

$$B(A) = \{lBIC_{0,n}(A) > lBIC_{1,n}(A)\}.$$

Case 1 ($K^* = K$):
If $K^* = K$, we already have the correct tree structure and we do not want to prune the tree any further. Therefore,

$$\begin{aligned} P(\mathcal{P}_n^{BIC} = \mathcal{P}_n^*) &\geq P(B(C_{K-1,n})) \\ &= \sum_{C_{K-1}^* \in \mathcal{C}_{K-1}} P(B(C_{K-1,n}) \cap \{C_{K-1,n} = C_{K-1}^*\}) \\ &= 1 + \sum_{C_{K-1}^* \in \mathcal{C}_{K-1}} \left[P(B(C_{K-1}^*) \cap \{C_{K-1,n} = C_{K-1}^*\}) - P(\{C_{K-1,n} = C_{K-1}^*\})\right]. \end{aligned}$$

From here, we split the sum into two parts. Consider the set S_1 of (non-random) regions in $\mathcal{C}_{K-1}$ that contain exactly one true split-point (no more and no less), the split-point is in the interior of the region, and the region is not the empty set. By a generalization of Lemma 2[1], it follows that for any fixed $C^*_{K-1} \in S_1$, $P(B(C^*_{K-1})) \to 1$. If it can be shown that $P(C_{K-1,n} \in S_1) \to 1$, then it follows from Lemma 3 that $P(\mathcal{P}_n^{BIC} = \mathcal{P}_n^*) \to 1$ in this case, because the above sum contains only finitely many terms. It therefore remains to show that $P(C_{K-1,n} \in S_1) \to 1$ as $n \to \infty$.

That $C_{K-1,n}$ contains exactly one split-point in the interior of the region with probability approaching one is given by (C6). It can also be seen that the probability of *at most* a single split-point approaches one by (C4) in combination with (C1). That the probability that the region is not the empty set approaches one is given by (C5). (Loosely speaking, these should all be satisfied if split-point estimators are weakly consistent for the location of the split-point and an appropriate grid $\mathcal{G}(\tilde{\epsilon}/N)$ is chosen.) Therefore, $P(C_{K-1,n} \in S_1) \to 1$ as $n \to \infty$ and this case is complete.

Case 2 ($K^* < K$):
Suppose now that $K^* < K$. This means that we need to prune the full tree to recover the true tree structure. We first show that the result holds for $K^* = K - 1$, from which it becomes clear that it is true for all $K^* < K$. For convenience, we denote the set $\{lBIC_{0,n}(A) < lBIC_{1,n}(A)\}$ by $B(A)^c$. (The complement would normally include the possibility of equality, but we do not need to refer to this possibility in any part of our proof.) When $K^* = K - 1$,

$$
\begin{aligned}
&P(\mathcal{P}_n^{BIC} = \mathcal{P}_n^*) \\
&\quad \geq P\left(B(C_{K-1,n})^c \cap B(C_{K^*-1,n})\right) \\
&\quad = \sum_{\substack{C^*_{K-1} \in \mathcal{C}_{K-1} \\ C^*_{K^*-1} \in \mathcal{C}_{K^*-1}}} P(B(C_{K-1,n})^c \cap \{C_{K-1,n} = C^*_{K-1}\} \cap B(C_{K^*-1,n}) \cap \{C_{K^*-1,n} = C^*_{K^*-1}\}).
\end{aligned}
$$

We now split the sum into four parts. Let S_1 in this case be the set of all (non-random) regions in $\mathcal{C}_{K-1}$ that contain no true split-points and the region is not the empty set. Let S_2 be the set of all (non-random) regions in $\mathcal{C}_{K^*-1}$ that contain exactly one true split-point (no more and no less), the split-point is in the interior of the region, and the region is not the empty set. The sum is split into four parts indexed by $S_1 \cap S_2$, $(\mathcal{C}_{K-1} \setminus S_1) \cap S_2$, $S_1 \cap (\mathcal{C}_{K^*-1} \setminus S_2)$, and $(\mathcal{C}_{K-1} \setminus S_1) \cap (\mathcal{C}_{K^*-1} \setminus S_2)$. We show that $P(C_{K-1,n} \in S_1, C_{K^*-1,n} \in S_2) \to 1$ to conclude that $P(\mathcal{P}_n^{BIC} = \mathcal{P}_n^*) \to 1$ in this case.

We start with $C_{K-1,n}$. The probability that this region contains no split-points approaches one by (C4) and (C1). That the set is nonempty with probability approaching one is given by (C5). Therefore, $P(C_{K-1,n} \in S_1) \to 1$.

[1]This is the generalization of Lemma 2 referred to after the statement of Theorem 1. In the generalized version of the lemma, the region A considered in the $lBIC$ and the support of $\mathbf{X}$ do not have to be the same (the support of $\mathbf{X}$ is allowed to be larger, so that we effectively discard a certain random proportion of the observations in the calculation of the $lBIC$). The proof should be very similar to the proof of Lemma 2, but it is left for future work.

For $C_{K^*-1,n}$, we see that the probability that the region contains exactly one split-point in the interior of the region approaches one by (C6). That the set is nonempty with probability approaching one is again given by (C5). Therefore, $P(C_{K^*-1,n} \in S_2) \to 1$.

From here, $P(C_{K-1,n} \in S_1, C_{K^*-1,n} \in S_2) \to 1$ and hence $P(\mathcal{P}_n^{BIC} = \mathcal{P}_n^*) \to 1$, completing the proof for both cases. □

A.4 Consistent Estimators of the Conditional Variance

We briefly explain why it is possible to replace the true (unknown) value of σ^2 with a consistent estimator $\hat{\sigma}_n^2$ in the BIC-ET algorithm and still consistently recover the true tree structure. Because traditional tree-construction algorithms do not depend on any inputs to the estimated or true value of σ^2 (they simply seek to minimize the sum of squares), what needs to be shown is that Lemma 2 holds when σ^2 is replaced with a consistent estimator, $\hat{\sigma}_n^2$.

Lemma 4. *If σ^2 is replaced with $\hat{\sigma}_n^2$ in the lBIC of Lemma 2, the same conclusions still hold provided that $\hat{\sigma}_n^2 \xrightarrow{p} \sigma^2$ and $\sigma^2 > 0$. (Note that the same estimate $\hat{\sigma}_n^2$ is used in all steps of the pruning algorithm.)*

Proof. For clarity, we write $lBIC_{d,n}([0,1]^p, \sigma^2)$ and $lBIC_{d,n}([0,1]^p, \hat{\sigma}_n^2)$ for $d = 0, 1$ to distinguish between the two cases (using the true and estimated value of σ^2, respectively). By assumption, $\hat{\sigma}_n^2 \xrightarrow{p} \sigma^2$ and so $1/\hat{\sigma}_n^2 \xrightarrow{p} 1/\sigma^2$ because $\sigma^2 > 0$.

From here, let $\epsilon = 0.2/\sigma^2$ (any sufficiently small ϵ will work). Now, $P(|1/\hat{\sigma}_n^2 - 1/\sigma^2| \leq \epsilon) \to 1$.

Case 1
We already know that with known σ^2,

$$P(lBIC_{1,n}([0,1]^p, \sigma^2) > lBIC_{0,n}([0,1]^p, \sigma^2)) \to 1.$$

Let

$$V = \max_{\substack{n_L=1,\ldots,n-1 \\ j=1,\ldots,p}} \left[\sum_{i=1}^{n} (Y_i - \bar{Y})^2 - \sum_{i=1}^{n_L} (Y_i^{*j} - \bar{Y}_{L,n_L}^{*j})^2 - \sum_{i=n_L+1}^{n} (Y_i^{*j} - \bar{Y}_{R,n_L}^{*j})^2 \right].$$

Note that $V \geq 0$. Then,

$$\begin{aligned}
&P(lBIC_{1,n}([0,1]^p, \hat{\sigma}_n^2) > lBIC_{0,n}([0,1]^p, \hat{\sigma}_n^2)) \\
&\quad \geq P\left(\left\{ \frac{1}{\hat{\sigma}_n^2} V < 3\log(n) \right\} \cap \{|1/\hat{\sigma}_n^2 - 1/\sigma^2| \leq \epsilon\} \right) \\
&\quad = P\left(\left\{ \left(\frac{1}{\sigma^2} + \frac{1}{\hat{\sigma}_n^2} - \frac{1}{\sigma^2} \right) V < 3\log(n) \right\} \cap \{|1/\hat{\sigma}_n^2 - 1/\sigma^2| \leq \epsilon\} \right) \\
&\quad \geq P\left(\left\{ \left(\frac{1}{\sigma^2} + \epsilon \right) V < 3\log(n) \right\} \cap \{|1/\hat{\sigma}_n^2 - 1/\sigma^2| \leq \epsilon\} \right) \\
&\quad = P\left(\left\{ \frac{1}{\sigma^2} V < 2.5\log(n) \right\} \cap \{|1/\hat{\sigma}_n^2 - 1/\sigma^2| \leq \epsilon\} \right) \\
&\quad \to 1,
\end{aligned}$$

because the probability of both events approaches one (the argument for the first event is the same as in Case 1 of Lemma 2). Note that as long as the same penalty is applied to $lBIC_{0,n}$ and $lBIC_{1,n}$ for the estimated variance, the penalty on the variance cancels in $lBIC_{0,n} - lBIC_{1,n}$.

Case 2

Let $D \geq 0$ be as in the proof of Lemma 2. Note that $P(N_L \notin \{0, n\}) \to 1$. From here,

$$\begin{aligned}
&P(lBIC_{0,n}([0,1]^p, \hat{\sigma}_n^2) > lBIC_{1,n}([0,1]^p, \hat{\sigma}_n^2)) \\
&\quad \geq P\left(\left\{\frac{1}{\hat{\sigma}_n^2} D > 3\log(n)\right\} \cap \{N_L \notin \{0,n\}\} \cap \{|1/\hat{\sigma}_n^2 - 1/\sigma^2| \leq \epsilon\}\right) \\
&\quad = P\left(\left\{\left(\frac{1}{\sigma^2} + \frac{1}{\hat{\sigma}_n^2} - \frac{1}{\sigma^2}\right) D > 3\log(n)\right\} \cap \{N_L \notin \{0,n\}\} \cap \{|1/\hat{\sigma}_n^2 - 1/\sigma^2| \leq \epsilon\}\right) \\
&\quad \geq P\left(\left\{\left(\frac{1}{\sigma^2} - \epsilon\right) D > 3\log(n)\right\} \cap \{N_L \notin \{0,n\}\} \cap \{|1/\hat{\sigma}_n^2 - 1/\sigma^2| \leq \epsilon\}\right) \\
&\quad = P\left(\left\{\frac{1}{\sigma^2} D > 3.75\log(n)\right\} \cap \{N_L \notin \{0,n\}\} \cap \{|1/\hat{\sigma}_n^2 - 1/\sigma^2| \leq \epsilon\}\right) \\
&\quad \to 1,
\end{aligned}$$

by the same argument as in Case 2 of Lemma 2. This completes the proof for both cases. □